robotics

MDPI

Article

Dynamic Space Debris Removal via Deep Feature Extraction and Trajectory Prediction in Robotic Systems

Zhuyan Zhang [1,2], Deli Zhang [1] and Barmak Honarvar Shakibaei Asli [2,*]

[1] College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; zhuyan.zhang.901@cranfield.ac.uk (Z.Z.); nuaazdl@126.com (D.Z.)
[2] Centre for Life-Cycle Engineering and Management, Faculty of Engineering and Applied Sciences, Cranfield University, Cranfield MK43 0AL, UK
* Correspondence: barmak@cranfield.ac.uk

Abstract

This work introduces a comprehensive vision-based framework for autonomous space debris removal using robotic manipulators. A real-time debris detection module is built upon the YOLOv8 architecture, ensuring reliable target localization under varying illumination and occlusion conditions. Following detection, object motion states are estimated through a calibrated binocular vision system coupled with a physics-based collision model. Smooth interception trajectories are generated via a particle swarm optimization strategy integrated with a 5–5–5 polynomial interpolation scheme, enabling continuous and time-optimal end-effector motions. To anticipate future arm movements, a Transformer-based sequence predictor is enhanced by replacing conventional multilayer perceptrons with Kolmogorov–Arnold networks (KANs), improving both parameter efficiency and interpretability. In practice, the Transformer+KAN model compensates the manipulator's trajectory planner to adapt to more complex scenarios. Each component is then evaluated separately in simulation, demonstrating stable tracking performance, precise trajectory execution, and robust motion prediction for intelligent on-orbit servicing.

Keywords: space debris; robotic arm serving; collision detection; particle swarm optimization; Kolmogorov–Arnold network

check for updates

Academic Editor: Dan Zhang

Received: 28 July 2025
Revised: 23 August 2025
Accepted: 25 August 2025
Published: 28 August 2025

Citation: Zhang, Z.; Zhang, D.; Honarvar Shakibaei Asli, B. Dynamic Space Debris Removal via Deep Feature Extraction and Trajectory Prediction in Robotic Systems. *Robotics* **2025**, *14*, 118. https://doi.org/10.3390/robotics14090118

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the increasing intensity of human space activities, the accumulation of space debris in Earth's orbit has become a pressing concern. According to the European Space Agency 2024 Space Environment Report, there are currently more than 35,000 pieces of debris larger than 10 cm in diameter orbiting the Earth, posing significant threats to operational satellites, space stations, and future missions [1], and the increasing risk of collisions has been a major challenge [2].

Traditional methods for removing debris, such as nets and harpoons, have seen various degrees of experimental success. Figure 1 shows a few examples of these devices. At the same time, robotic manipulator servicing systems provide a novel approach to capture orbital debris, as illustrated in Figure 2. Unlike the other two systems, robotic arms are considered particularly promising because of their flexibility and reusability [3]. However, the dynamic and irregular motion of space debris requires advanced solutions that involve real-time multitarget detection, motion prediction, and intelligent path planning [4]. These capabilities are essential to ensure safety and efficiency during autonomous capture operations.

Failure to address space debris risks could result in catastrophic collisions, potentially triggering the Kessler Syndrome [5]—a chain reaction of debris-generating events that could jeopardize the usability of entire orbital regions. Therefore, developing intelligent robotic systems with autonomous visual servoing and adaptive control capabilities is a technical challenge and a critical safeguard for sustainable space development.

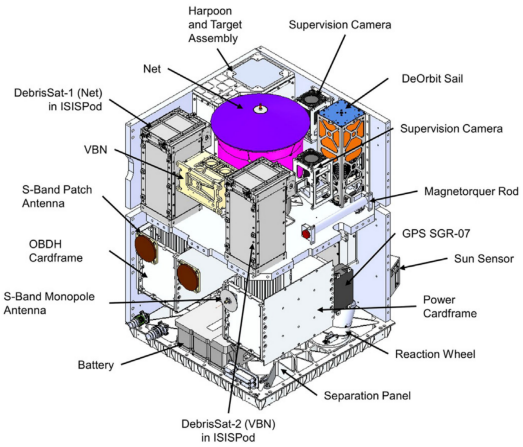


Figure 1. Devices for capturing space debris using nets and harpoons [6].

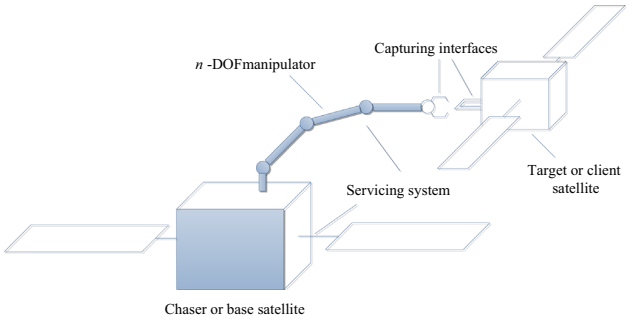


Figure 2. Devices for capturing space debris using a robotic arm [7].

2. Related Work

With the increasing complexity of autonomous space debris removal missions, the integration of advanced perception, prediction, and control technologies has become essential. This chapter provides a comprehensive review of recent developments in four key areas relevant to dynamic debris removal by robotic systems. Section 2.1 examines the current status of space debris collection technologies, including various capture mechanisms. Section 2.2 reviews advances in object feature extraction methods, focusing on deep learning-based visual perception. Section 2.3 discusses state-of-the-art approaches for object motion prediction, highlighting both model-based and data-driven techniques. Finally, Section 2.4 summarizes progress in robotic arm path planning, emphasizing efficient trajectory optimization in dynamic environments. Together, these reviews lay the foundation for the integrated framework proposed in this study.

2.1. Space Debris Collection

The European Space Agency (ESA) proposed the "RemoveDEBRIS" project, which tested various innovative space debris removal technologies, including net capture and harpoons, as shown in Figures 3 and 4. These technologies have achieved preliminary results in laboratory and orbital simulation environments [6], but challenges remain in capturing fast-moving, tumbling debris in real orbital environments, particularly under multitarget dynamic coordination scenarios.

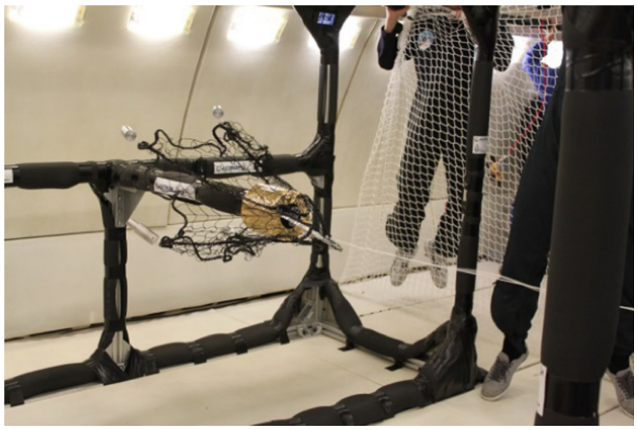


Figure 3. Device for capturing orbital debris using a net [6].

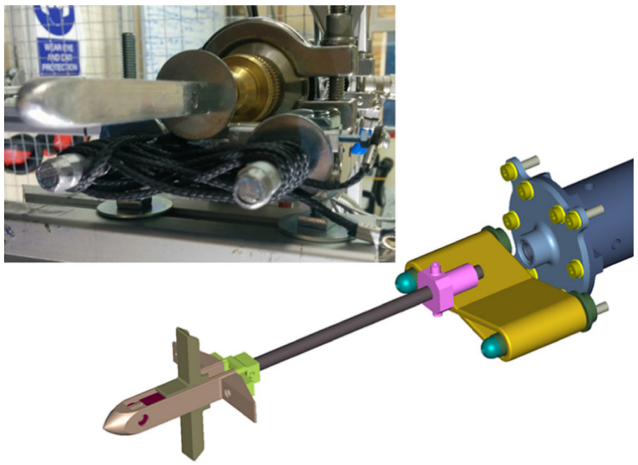


Figure 4. Device for capturing orbital debris using harpoons [6].

Robotic arm-based approaches have received increasing attention due to their controllability and reusability. The US DARPA initiated the Robotic Servicing of Geosynchronous Satellites (RSGS) project, aiming to use dexterous manipulators for on-orbit servicing and active debris removal tasks [8]. While such approaches have made progress, challenges persist in path planning, especially under dynamic and multi-body interactions. Beyond experimental and conceptual systems, research on autonomous robotic arms has proposed various modeling and control frameworks to address motion planning in cluttered, dynamic environments. For instance, Mu et al. [9] presented a unified model of multiple moving obstacles and introduced a pseudo-distance-based trajectory optimization method for redundant space manipulators. This enables real-time collision avoidance in environments populated with high-speed debris.

Several studies have addressed the challenges of controlling space robots for active debris removal. Dubanchet et al. [10] explored the dynamic modeling and control strategies for such systems, emphasizing trajectory tracking performance under dynamic coupling and limited onboard computation. In particular, Dubanchet's work is instrumental in addressing buffer compliance control by integrating compliant actuators and force sensors to manage impacts during debris capture, thereby improving the system's ability to handle uncertain, dynamic environments in space. Recent studies have proposed more robust impedance control strategies. For example, Palma et al. [11] demonstrated the application of model-based impedance control for free-floating space manipulators during debris capture tasks, where the manipulator adapts its dynamics to safely interact with tumbling debris, minimizing joint torque and system disturbance during contact. This model-based

approach ensures a smoother interaction compared to classical Cartesian controllers, particularly under dynamic loading conditions such as with non-cooperative targets. In contrast, Sampath and Feng [12] concentrated on enhancing the motion control of space manipulators by integrating neural networks with sliding mode and computed torque controllers, thereby improving the joint tracking accuracy and robustness to system uncertainties. These studies collectively underscore the importance of robust and adaptive manipulator control in achieving reliable debris capture under real-world orbital conditions.

In addition to mission-specific robotic arms developed for active debris removal, general-purpose space manipulators also provide valuable insights. One representative example is the European Robotic Arm (ERA), developed by ESA for space station servicing tasks such as payload transfer and maintenance (see Figure 5). Although the ERA was not originally designed for high-speed debris capture, its symmetrical structure, high payload capacity, and multi-DOF configuration demonstrate engineering principles that are potentially transferable to future on-orbit debris handling systems [13].

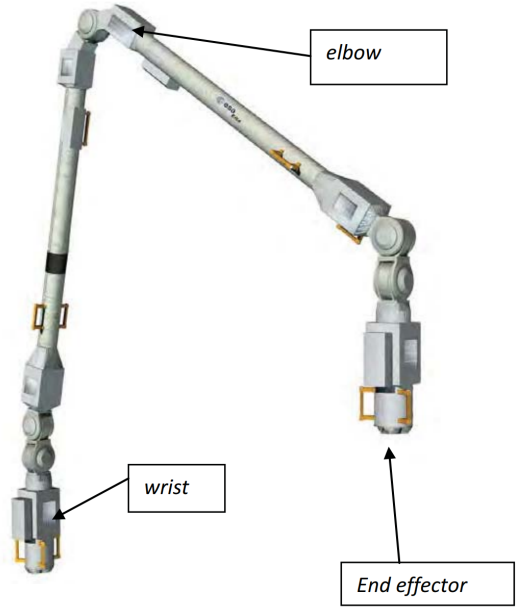


Figure 5. The European Robotic Arm [13].

To provide a clearer comparison, Table 1 summarizes the main technologies for space debris removal discussed above, including their principles and major limitations.

Table 1. Summary of main space debris removal technologies.

Method	Challenges
Net [6]	Difficult to target fast, dynamic debris; limited adaptability
Harpoon [6]	Risk of fragmentation; targeting accuracy concerns
Robotic Arm [8,9,13–15]	Complex trajectory planning; real-time perception-control integration; dynamic coordination issues

2.2. Object Feature Extraction Technology

Recent advances in deep learning and computer vision have made integrating these models with visual servoing a key research focus. However, traditional methods still struggle with target detection and tracking in dynamic environments, especially in the presence of occlusion, lighting changes, and small targets. Improving the efficiency and accuracy of these tasks using deep learning models is a primary challenge in visual servoing.

To address these challenges, researchers have proposed various solutions. In 2018, Zhang et al. introduced ShuffleNet [16], a lightweight convolutional neural network that can efficiently run on mobile devices and embedded platforms, providing support for scenarios with limited computational resources.

In 2019, Carion et al. introduced DETR (Detection Transformer), a Transformer-based object detection model that performs end-to-end detection by leveraging self-attention to capture global image features. While it significantly improves detection accuracy and robustness, its high computational cost limits its suitability for real-time visual servoing tasks [17].

In 2020, Zhu et al. proposed Deformable DETR, an improved version of the DETR model that significantly reduces computational cost by introducing a sparse attention mechanism. This approach allows the model to focus on a limited number of relevant sampling points, improving its ability to detect small and densely located targets. These enhancements make it more suitable for real-time detection tasks in dynamic environments, such as space debris monitoring [18].

A comparative overview of prominent object feature extraction methods used in visual servoing is presented in Table 2, highlighting their technical characteristics and respective advantages. In recent years, considerable efforts have been directed towards incorporating lightweight spatio-temporal feature extraction networks into visual servoing frameworks to enhance system responsiveness and adaptability. These developments mark important progress in overcoming the constraints of conventional approaches, enabling the more accurate and efficient target detection and tracking.

Table 2. Summary of main object feature extraction methods in visual servoing.

Method	Key Characteristics	Advantages
ShuffleNet [16]	Lightweight convolutional neural network with channel shuffle operation	Efficient, low computational cost, suitable for mobile/embedded platforms
DETR [17]	Transformer-based end-to-end object detection with self-attention mechanism	Captures global features, high detection accuracy, robust to occlusion
Deformable DETR [18]	Introduces deformable attention for adaptive receptive fields	Better detection for small objects, reduced computational overhead, improved efficiency

2.3. Object Motion State Prediction

Trajectory prediction has become a key research area, but challenges remain in complex environments. Model-based methods struggle with generalization, while deep learning methods need further optimization for real-time performance and adaptability. Researchers are working on solutions to improve prediction accuracy and robustness.

Model-based methods establish physical models of the target, using features such as speed, acceleration, force, and position for modeling. Chen et al. captured the trajectory of a ping-pong ball and constructed an aerodynamic model to predict its future position [19]. The trajectory prediction results are shown in Figure 6. Qiao Shaojie improved traditional mathematical models using hidden Markov models to solve accuracy problems in traffic flow trajectory simulation [20]. Wiest et al. combined Gaussian mixture models with Bayesian methods to improve the generalization and accuracy of predictions [21].

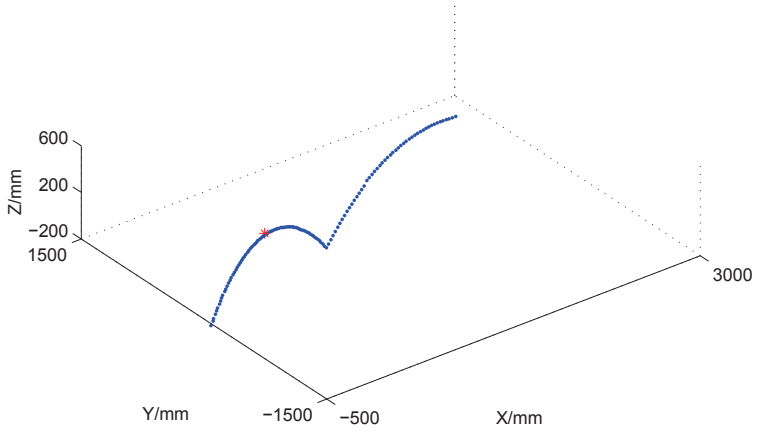


Figure 6. A sample trajectory and the predicted result. The dashed points are trajectory points, and the star is the predicted hitting point [19].

Deep learning-based methods rely on neural networks to autonomously learn motion features and perform well in complex trajectory prediction tasks. Although traditional RNNs are widely used for temporal information processing, their performance is limited due to the vanishing gradient problem, leading to the development of Long Short-Term Memory (LSTM) networks to address this challenge. Alahi et al. introduced a social pooling layer in Social LSTM to capture interactions between pedestrians and successfully predict nonlinear behaviors [22].

Gupta combined LSTM with Generative Adversarial Networks (LSTM-GANs) and optimized pedestrian trajectory prediction using a pooling mechanism [23]. The model integrates past trajectories, encodes human–human interactions via a pooling module, and generates socially acceptable future trajectories through adversarial training.

Shafiee proposed a social attention model that integrates LSTM with attention mechanisms to effectively reduce prediction errors [24]. The model uses a conditional 3D attention mechanism to extract dynamic social and environmental cues from video input, enhancing trajectory prediction through spatio-temporal awareness.

Table 3 compiles representative approaches for predicting object motion states, outlining their fundamental principles and key advantages. These methods form the theoretical and technical basis for the advancement of trajectory prediction in dynamic environments. The insights gained from such comparisons contribute to addressing challenges related to robustness, adaptability, and real-time performance in complex motion scenarios.

Table 3. Summary of main object motion state prediction methods.

Method	Strengths/Advantages
Physics-based Model [19].	Good interpretability, effective for simple environments
HMTP [20].	Handles sequential dependencies, improves prediction in dynamic scenarios
GMM [21].	Captures multi-modal uncertainty, better generalization
LSTM-based DL [22–24].	Learns complex patterns, handles nonlinear/interactive behaviors, strong in complex, real-world tasks

2.4. Robotic Arm Path Planning

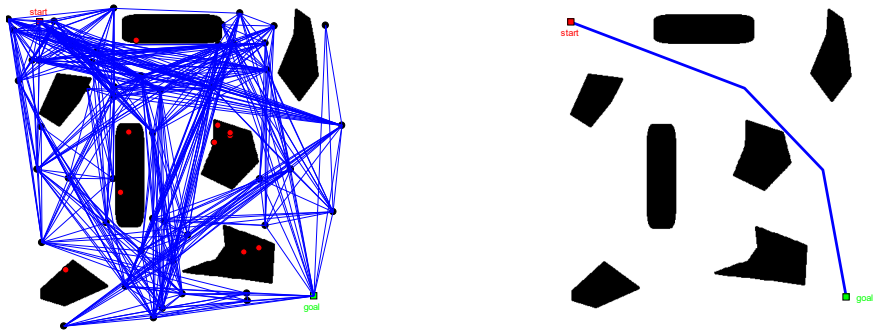
Recent advancements in robotics have made robotic arm path planning a key research focus. However, achieving efficient and precise path planning in dynamic, complex environments remains challenging due to limitations in traditional methods and the need for further improvements in modern algorithms, particularly in real-time performance, robustness, and resource efficiency.

Traditional path planning methods mainly include geometry-based and optimization-based algorithms. For example, Rapidly Exploring Random Tree (RRT) is a classic sampling-based path planning algorithm. Kuffner et al. proposed that it can quickly explore free spaces in high-dimensional spaces but generates paths with poor smoothness, limiting its practical application [25]. To address this issue, Karaman et al. introduced the RRT* algorithm [26], which optimizes path smoothness through postprocessing, but its computational efficiency is insufficient in complex dynamic environments.

Another method is Model Predictive Control (MPC), widely used for industrial robot path optimization tasks [27]. This method optimizes paths by predicting future states in real-time, but is less adaptable to complex obstacle environments.

Modern intelligent algorithms have injected new vitality into path planning research. For example, Reinforcement Learning (RL) can learn efficient path planning strategies in dynamic environments through interactive training [28].

Neural network-based path generation methods have also gradually become a research focus. Hsu et al. proposed a probabilistic roadmap (PRM) method combined with deep neural networks, using neural networks to learn environmental characteristics and generate paths [29]. As shown in Figure 7, the PRM algorithm constructs a roadmap by randomly sampling collision-free points and connecting them to plan a feasible path from start to goal while avoiding obstacles.



(**a**) (**b**)

Figure 7. Illustration of the PRM algorithm: random sampling and roadmap construction for path planning, (**a**) roadmap and (**b**) planned path [30].

This significantly improved the path planning efficiency, but its adaptability in dynamic environments still needs to be improved. Furthermore, Marini et al. mentioned that Particle Swarm Optimization (PSO) uses both global and local search methods to optimize paths in dynamic environments, completing robot obstacle avoidance and optimal path generation. PSO has advantages such as strong global optimization capability, simple algorithm implementation, and high computational efficiency, as shown in Figure 8 [31].

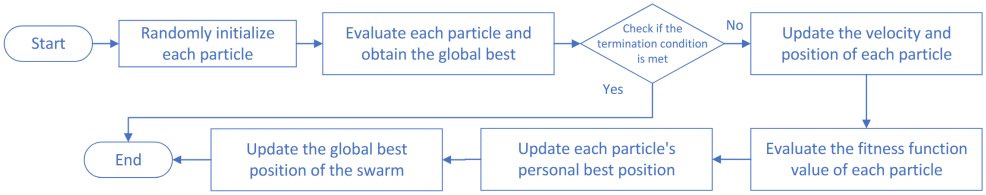


Figure 8. PSO algorithm flow [32].

Lillicrap discussed DDPG (Deep Deterministic Policy Gradient), which combines the strengths of deep learning and reinforcement learning. It uses an Actor–Critic architecture to efficiently optimize strategies in continuous action spaces and is widely used in path

planning and decision-making tasks for complex dynamic scenarios such as robot control and autonomous driving, providing an effective method for solving high-dimensional continuous control problems [33].

Various approaches to planning the path of the main robotic arm are compared in Table 4, each characterized by different algorithms and performance advantages. Despite notable advancements, achieving efficient and reliable planning in dynamic and uncertain environments remains an open challenge. Research is increasingly directed towards improving adaptability, fusing heterogeneous sensor data for informed decision making, and designing lightweight algorithms suitable for real-time robotic applications.

Table 4. Summary of main robotic arm path planning methods.

Method	Strengths/Advantages
RRT/RRT* [25,26]	Efficient in high-dimensional spaces, suitable for rapid exploration; RRT* improves path optimality and smoothness
MPC [27]	Real-time path optimization, considers system dynamics and constraints
RL [28]	Adaptive to dynamic environments, can handle complex tasks via trial-and-error learning
PRM [29]	Efficient roadmap construction, suitable for static environments
PSO [31]	Strong global optimization ability, simple implementation, high efficiency
DDPG [33]	Effective in high-dimensional continuous control, good for complex dynamic tasks

3. Methodology

3.1. Datasets

This study utilizes two distinct datasets, each tailored to a specific task relevant to space operations and robotic control. The first dataset is designed for the detection of space debris, addressing a pressing challenge in orbital safety. The second dataset supports the prediction of robotic arm trajectories, focusing on learning dynamic motion patterns from optimized path planning data. Together, these datasets provide a comprehensive foundation for developing robust machine learning models in two critical application domains: space environment perception and autonomous robotic movement.

3.1.1. Space Debris Detection Dataset

This dataset is centered on the problem of detecting space debris, which is essential for ensuring the safety and sustainability of satellite and orbital missions. It contains a diverse array of synthetically generated images that mimic the appearance of meteoroids and fragmented debris in outer space. These images depict objects of varying shapes, sizes, and trajectories set against realistic space backgrounds.

Each image is rigorously annotated, including the bounding box coordinates and exact positions of all visible debris. These annotations are critical for training and evaluating object detection algorithms under challenging orbital conditions. The dataset introduces high variability in terms of illumination, object motion, and background noise, offering a demanding testbed for detection models. Figure 9 illustrates a representative image from the dataset, highlighting the visual difficulty posed by small, irregularly shaped, and low-contrast debris against a dark space environment.

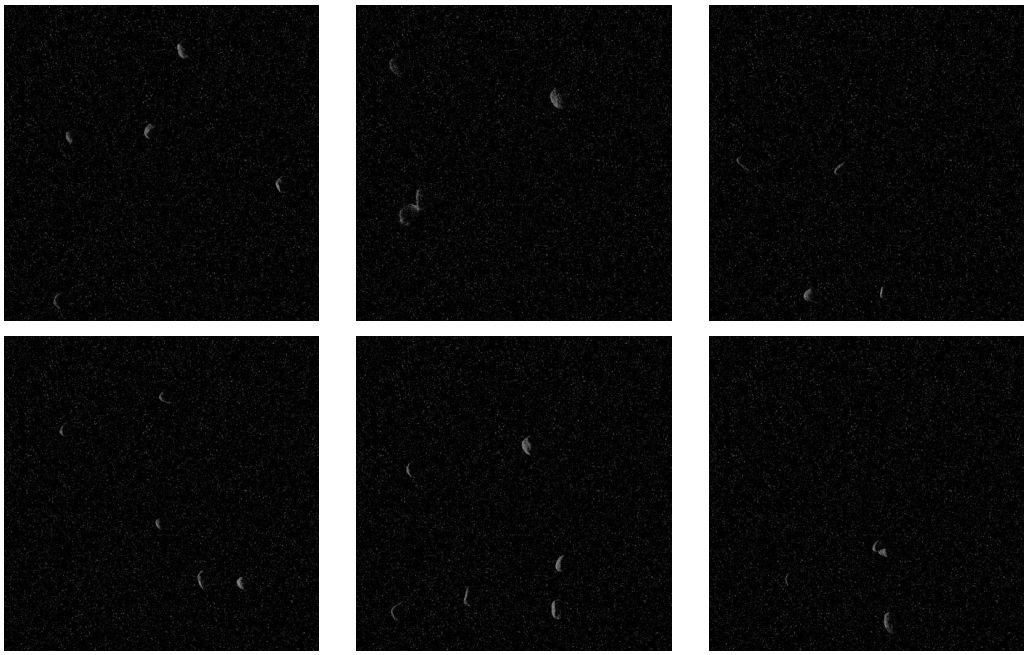


Figure 9. Dataset sample images—from https://www.aicrowd.com/challenges/ai-blitz-7/problems/debris-detection#dataset, accessed on 19 May 2025. The images vary in debris density, shape, brightness, and spatial arrangement. High background noise and low object visibility create significant challenges for accurate detection and localization.

3.1.2. Robotic Arm Trajectory Dataset

The second dataset is constructed using the particle swarm optimization (PSO) algorithm to generate optimized motion trajectories for a robotic arm. The PSO algorithm ensures that the planned trajectories are both smooth and physically feasible, reflecting a broad spectrum of dynamic conditions encountered in robotic control [34].

These trajectory samples are used to train the Transformer-KAN network, which is tasked with learning and accurately forecasting the motion of the robotic arm. This dataset plays a vital role in capturing the nonlinear and temporal dependencies of robotic movements, thus enabling the prediction model to generalize across complex motion patterns. The inclusion of diverse trajectory profiles enhances the model's ability to perform robustly in real-world scenarios requiring high-precision control and adaptive planning.

3.2. Advantages and Structure of YOLOv8 for Space Debris Detection

Space debris presents unique challenges for visual detection systems due to its diverse and dynamic nature. The characteristics of space debris vary widely, encompassing objects ranging from small fragments of satellites and rockets to larger defunct spacecraft. These objects differ in shape, size, material, and reflectivity, often exhibiting irregular surfaces that can make detection difficult. Moreover, space debris moves at high velocities, often in orbits that are hard to track consistently. The dynamic motion of debris, combined with factors such as occlusion, varying lighting conditions, and the vast size of the orbital space, introduces significant complexity for any visual recognition system [5,6].

Given these challenges, selecting the right model for detecting space debris is crucial. YOLOv8, a state-of-the-art object detection architecture, is particularly well-suited for this task. YOLOv8 is renowned for its ability to perform real-time, high-accuracy object detection, making it ideal for the high-speed and large-scale environment of space. Its capability to process large amounts of visual data quickly and efficiently enables it to detect objects, even in the presence of occlusion or inconsistent lighting conditions [35]. These

advantages make YOLOv8 a powerful tool for improving the accuracy and reliability of space debris detection in a constantly changing environment.

While earlier versions of YOLO, such as YOLOv5 [36], YOLOv6 [37], and YOLOv7 [38], have demonstrated strong performance in various object detection tasks, they each come with limitations when applied to the complex and high-speed environment of space debris detection. YOLOv5, for instance, although efficient and lightweight, lacks some of the architectural refinements introduced in YOLOv8, such as dynamic input scaling and a decoupled head for classification and localization. YOLOv6 introduced improvements for industrial scenarios, but still falls short in terms of generalization to unstructured and dynamic scenes like outer space. YOLOv7, while notable for introducing E-ELAN and model reparameterization strategies, does not match the inference stability and modularity that YOLOv8 offers.

Moreover, YOLOv8 integrates several enhancements, including a unified backbone, advanced anchor-free mechanisms, and improved feature fusion strategies, making it more adaptable to tasks involving small, fast-moving, and occluded objects—key characteristics of space debris. These considerations led us to adopt YOLOv8 as the most suitable detection backbone for this study.

YOLOv8 is a single-stage, real-time object detection model composed of an input preprocessing module, a backbone network, a neck for feature fusion, and an output prediction head. The input pipeline enhances generalization through mosaic augmentation, adaptive anchor computation, and grayscale padding. These steps help the model to deal with complex visual variations in real-world scenes.

The backbone of YOLOv8 processes input images through a series of convolutional layers and C2f (cross-stage partial fusion) modules to extract multi-scale features with high efficiency. Compared to the C3 block used in YOLOv5/YOLOv7, the C2f module introduces a simplified and more effective residual design, inspired by the ELAN architecture [38].

Specifically, it removes a convolutional layer while leveraging multiple Bottleneck blocks to facilitate rich feature learning and enhance gradient propagation.

The detailed architecture of the C2f module is illustrated in Figure 10. As shown, the input tensor is split into several partitions, each independently processed by lightweight residual blocks. These outputs are then concatenated and passed through a final convolutional unit (CBS block) to fuse the features, achieving both parameter efficiency and expressive power.

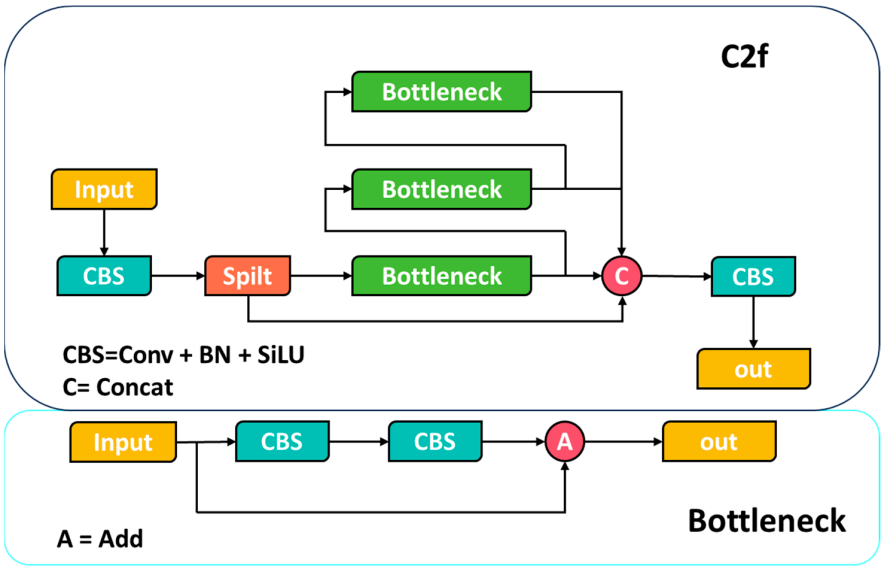


Figure 10. Structural diagram of the C2f module [35].

The output features of the backbone are passed through the Spatial Pyramid Pooling-Fast (SPPF) module. This module applies max-pooling operations with different kernel sizes to aggregate contextual information from multiple receptive fields:

$$\mathrm{SPPF}(x) = \mathrm{Concat}\left(\mathrm{MaxPool}_{5\times5}(x), \mathrm{MaxPool}_{9\times9}(x), \mathrm{MaxPool}_{13\times13}(x), x\right), \tag{1}$$

where x denotes the input feature map produced by the backbone (shape $C \times H \times W$). $\mathrm{MaxPool}_{k\times k}(\cdot)$ is a max-pooling operation with kernel size $k \times k$, stride 1, and appropriate padding to preserve spatial dimensions, used to aggregate context under different receptive fields. $\mathrm{Concat}(\cdot)$ concatenates the resulting tensors along the channel dimension. $\mathrm{SPPF}(x)$ is the output of the Spatial Pyramid Pooling—Fast module; if no additional channel compression is applied, its channel count becomes $4C$ by stacking the three pooled features and the original input. The kernel sizes $k \in \{5, 9, 13\}$ are chosen to capture multi-scale contextual information.

The neck design of YOLOv8 follows a combination of Feature Pyramid Network (FPN) [39] and Path Aggregation Network (PAN) [40], which fuse high-level semantic and low-level spatial information across scales. This multi-scale fusion structure improves object localization and classification at various object sizes.

In the head, YOLOv8 separates classification and regression into decoupled branches. The classification branch uses binary cross-entropy (BCE) loss, and the regression branch uses distribution focal loss (DFL) [41] combined with CIoU loss. YOLOv8 employs the TaskAligned Assigner [42] to dynamically assign positive samples by calculating a joint alignment score:

$$\mathcal{S}_{\mathrm{align}} = \sigma(p_{\mathrm{cls}}) \cdot \mathrm{IoU}(b_{\mathrm{pred}}, b_{\mathrm{gt}}), \tag{2}$$

where p_{cls} denotes the predicted (pre-sigmoid) classification logit for the target class, $\sigma(\cdot)$ is the sigmoid function mapping it into $(0, 1)$, $\mathrm{IoU}(b_{\mathrm{pred}}, b_{\mathrm{gt}})$ is the intersection-over-union between the predicted bounding box b_{pred} and the ground-truth box b_{gt}, and IoU is defined as

$$\mathrm{IoU} = \frac{|A \cap B|}{|A \cup B|}, \tag{3}$$

with A and B the planar regions covered by b_{pred} and b_{gt}, $|A \cap B|$ their overlapping area, and $|A \cup B|$ their union area (overlap counted once). Thus, $\mathrm{IoU} \in [0, 1]$ quantifies localization quality, and the product form of $\mathcal{S}_{\mathrm{align}}$ ensures high scores only for boxes that are both confidently classified and well aligned.

An object is considered correctly detected if $\mathrm{IoU} > 0.5$. This joint optimization of classification and localization helps reduce misalignment in prediction and improves detection robustness.

Overall, the YOLOv8 model's efficiency and accuracy make it well-suited for real-time object detection in challenging environments, such as detecting and tracking space debris, where the timely and precise localization of multiple targets is essential.

3.3. Space Debris Trajectory Prediction Model

3.3.1. Velocity Extraction via Binocular Vision

To estimate object velocities in 3D space, we utilize a calibrated binocular vision system [43] comprising two synchronized cameras with parallel optical axes and a fixed baseline B (see Figure 11 and Table 5) . The velocity extraction pipeline consists of stereo object localization, 3D position reconstruction, temporal differentiation, and kinematic validation.

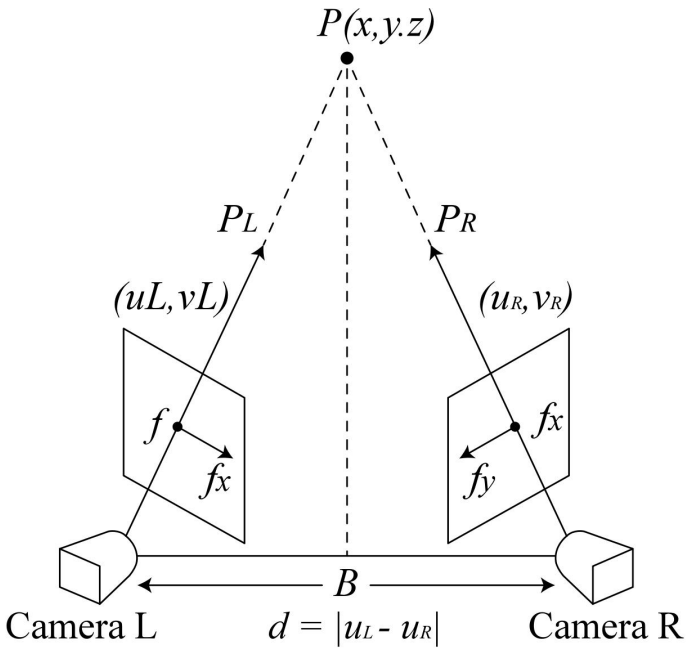


Figure 11. Binocular vision system geometry.

The stereo cameras provide synchronized image streams $I_L(t)$ and $I_R(t)$ satisfying the epipolar constraint. After distortion correction, target positions in the left and right images are $\mathbf{p}_L = (u_L, v_L)^T$ and $\mathbf{p}_R = (u_R, v_R)^T$. The horizontal disparity is $d_k = |u_L - u_R|$.

The 3D position $\mathbf{r}_k = (x_k, y_k, z_k)^T$ in camera coordinates is computed as

$$z_k = \frac{f_x \cdot B}{d_k}, \tag{4}$$

$$x_k = \frac{z_k \cdot (u_L - c_x)}{f_x}, \tag{5}$$

$$y_k = \frac{z_k \cdot (v_L - c_y)}{f_y}, \tag{6}$$

For consecutive frames at t_k and t_{k+1}, the instantaneous velocity vector is

$$\mathbf{v}_k = \frac{\mathbf{r}_{k+1} - \mathbf{r}_k}{\Delta t}, \quad \Delta t = t_{k+1} - t_k, \tag{7}$$

A sliding-window average is used to reduce the measurement noise:

$$\bar{\mathbf{v}} = \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{\mathbf{r}_{i+1} - \mathbf{r}_i}{\Delta t} \quad (N = 5), \tag{8}$$

Kinematic validation is enforced by bounding acceleration $\|\mathbf{v}_{k+1} - \mathbf{v}_k\| \leq a_{\max} \Delta t$ and by position prediction $\hat{\mathbf{r}}_{k+1} = \mathbf{r}_k + \bar{\mathbf{v}} \Delta t$ within a threshold $\delta_{\max}$.

Finally, velocities are transformed to world coordinates:

$$\mathbf{v}_{\text{world}} = \mathbf{R}^T \mathbf{v}_{\text{camera}}. \tag{9}$$

The extracted velocities $\mathbf{v}_1$ and $\mathbf{v}_2$ serve as inputs for downstream tasks such as collision prediction, post-collision dynamics, and trajectory simulation. The definitions of the parameters used above are summarized in Table 5. The system configuration and measurement settings are given in Table 6.

Table 5. Binocular vision parameters and definitions.

Symbol	Description
f_x, f_y	Focal lengths in x and y directions (pixels)
c_x, c_y	Principal point coordinates in the image plane (pixels)
B	Baseline, distance between camera centers (meters)
u_L, v_L	Pixel coordinates of the target in the left image (undistorted)
u_R, v_R	Pixel coordinates of the target in the right image (undistorted)
d_k	Disparity, $\lvert u_L - u_R \rvert$ (pixels)
z_k	Depth (distance from baseline to target, meters)
x_k, y_k	Lateral and vertical positions in the camera coordinates (m)
$\mathbf{r}_k$	3D position vector at frame k
$\mathbf{v}_k$	Instantaneous velocity vector at frame k
Δt	Time interval between consecutive frames (s)
$\mathbf{R}$	Rotation matrix (camera to world coordinates)
$a_{\max}$	Acceleration upper bound (m/s^2)
$\delta_{\max}$	Position prediction threshold (m)

Table 6. Stereo system configuration and measurement settings.

Parameter	Value
Frame rate	30 Hz (Δt = 33 ms)
Baseline B	0.5 m
Spatial accuracy	$\pm$1.5 cm
$a_{\max}$	9.8 m/s^2
$\delta_{\max}$	0.1 m

3.3.2. Kinematic Model of Spatial Binary-Sphere Collision

This section describes the kinematic model for the collision of two spheres in 3D space. At the instant of collision, the positions of the two spheres are given by (see Figure 12)

$$\mathbf{r}_1^* = \mathbf{r}_1^0 + \mathbf{v}_1 t^*, \tag{10}$$

$$\mathbf{r}_2^* = \mathbf{r}_2^0 + \mathbf{v}_2 t^*, \tag{11}$$

where t^* is the collision time, and $\mathbf{r}_i^0$ and $\mathbf{v}_i$ are the initial positions and velocities of the spheres ($i = 1, 2$) (see Table 7).

The collision normal vector is defined as

$$\mathbf{n} = \frac{\mathbf{r}_1^* - \mathbf{r}_2^*}{\|\mathbf{r}_1^* - \mathbf{r}_2^*\|}. \tag{12}$$

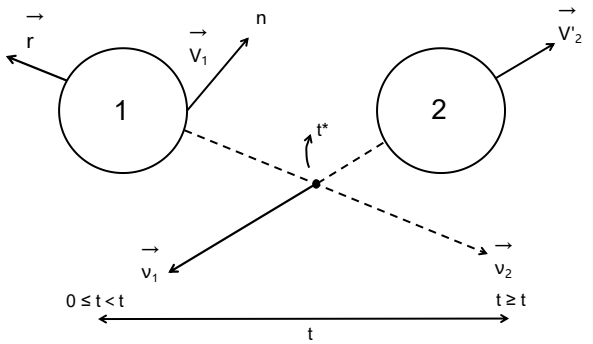


Figure 12. Schematic of binary-sphere collision, showing initial and collision positions and velocities, normal vector, and post-collision velocities.

The post-collision velocities are governed by the principles of conservation of momentum and, for elastic collisions, kinetic energy [44]. For a perfectly elastic collision:

$$\mathbf{v}_1' = \mathbf{v}_1 - \frac{2m_2}{m_1 + m_2}[(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}]\,\mathbf{n}, \tag{13}$$

$$\mathbf{v}_2' = \mathbf{v}_2 + \frac{2m_1}{m_1 + m_2}[(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}]\,\mathbf{n}. \tag{14}$$

For inelastic collisions, a restitution coefficient e is introduced:

$$e = -\frac{(\mathbf{v}_1' - \mathbf{v}_2') \cdot \mathbf{n}}{(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}}. \tag{15}$$

The post-collision velocities then become

$$\mathbf{v}_1' = \mathbf{v}_1 - \frac{(1+e)m_2}{m_1 + m_2}[(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}]\,\mathbf{n}, \tag{16}$$

$$\mathbf{v}_2' = \mathbf{v}_2 + \frac{(1+e)m_1}{m_1 + m_2}[(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}]\,\mathbf{n}. \tag{17}$$

The piecewise representation of each sphere'strajectory is

$$\mathbf{r}_i(t) = \begin{cases} \mathbf{r}_i^0 + \mathbf{v}_i t, & 0 \le t < t^*, \\ \mathbf{r}_i^* + \mathbf{v}_i'(t - t^*), & t \ge t^*. \end{cases} \tag{18}$$

(The parameters used above are summarized in Table 7).

Table 7. Parameter definitions for binary-sphere collision dynamics.

Symbol	Description
$\mathbf{r}_i^0$	Initial position of sphere i ($i = 1, 2$)
$\mathbf{v}_i$	Initial velocity of sphere i ($i = 1, 2$)
t^*	Time at which collision occurs
$\mathbf{r}_i^*$	Position of sphere i at the collision instant
$\mathbf{n}$	Unit normal vector at the point of contact
m_1, m_2	Masses of spheres 1 and 2
$\mathbf{v}_i'$	Post-collision velocity of sphere i
e	Coefficient of restitution ($0 \le e \le 1$)
$\mathbf{r}_i(t)$	Position of sphere i at time t

For multiple spheres, event-driven simulation and pairwise collision detection are required. Additional model complexity can be considered by including angular momentum, rotational motion, friction, or energy dissipation effects.

3.4. Robotic Arm Trajectory Model

3.4.1. Particle Swarm Optimization and 5–5–5 Polynomial Interpolation Trajectory Planning

In this study, the particle swarm optimization (PSO) algorithm is employed to optimize the motion trajectories of a robotic manipulator. PSO is a population-based stochastic algorithm inspired by the collective behavior observed in biological systems such as bird flocking and fish schooling [34]. It has proven effective for solving high-dimensional, continuous optimization problems and is particularly suitable for motion planning tasks due to its simplicity and strong global search capability [45].

Compared to classical search algorithms like A*, which is tailored for discrete graph traversal, or RRT, which is ideal for complex environments with dynamic obstacles, PSO is more efficient in structured or open spaces where obstacle avoidance is not critical. Its ability to converge rapidly towards optimal solutions makes it especially suitable for inverse kinematics and smooth trajectory generation.

To represent the robot's trajectory, we adopt a 5–5–5 polynomial interpolation method. The complete motion is divided into three phases: from the starting point to the first midpoint using a fifth-order polynomial, from the first midpoint to the second using another fifth-order polynomial, and from the second midpoint to the endpoint using a third fifth-order polynomial [46]. The position of each joint i during each segment is expressed as

$$\begin{aligned} l_1^{(i)}(t) &= a_{15}t^5 + a_{14}t^4 + a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10}, \\ l_2^{(i)}(t) &= a_{25}t^5 + a_{24}t^4 + a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20}, \\ l_3^{(i)}(t) &= a_{35}t^5 + a_{34}t^4 + a_{33}t^3 + a_{32}t^2 + a_{31}t + a_{30}, \end{aligned} \tag{19}$$

where the coefficients a_{ij} are determined by satisfying continuity and boundary conditions at the interpolation points. These conditions ensure that velocity and acceleration remain continuous throughout the trajectory.

In PSO, each solution candidate is treated as a particle characterized by its position and velocity. The position x_i represents the current joint configuration, while the velocity v_i determines the update direction. Each particle retains knowledge of its best-found position $p_{\text{best},i}$ and also has access to the globally best-known solution g_{best} found by the swarm. The velocity and position of each particle are updated at each iteration as follows:

$$\begin{aligned} v_i^{(t+1)} &= \omega v_i^{(t)} + c_1 r_1 \left(p_{\text{best},i} - x_i^{(t)} \right) + c_2 r_2 \left(g_{\text{best}} - x_i^{(t)} \right), \\ x_i^{(t+1)} &= x_i^{(t)} + v_i^{(t+1)}, \end{aligned} \tag{20}$$

where $r_1, r_2 \sim \text{Uniform}(0,1)$ are random factors, and c_1, c_2 are acceleration coefficients, typically set to 2. To avoid divergence, the velocity is limited by a maximum threshold $V_{\max}$, ensuring $\|v_i\| \leq V_{\max}$.

To further improve convergence behavior, a linearly decreasing inertia weight strategy is adopted. The inertia weight ω starts at a high value and gradually decreases to favor exploitation over exploration. This strategy, known as the Linearly Decreasing Weight (LDW) approach, is defined as

$$\omega^{(t)} = (\omega_{\text{ini}} - \omega_{\text{end}}) \cdot \left(\frac{G_k - g}{G_k} \right) + \omega_{\text{end}}, \tag{21}$$

where ω_{ini} and ω_{end} are the initial and final inertia weights, G_k is the total number of iterations, and g is the current iteration number.

The overall workflow of the PSO algorithm is illustrated in Figure 13, which highlights the steps from initialization through iterative evaluation and update processes until convergence criteria are met.

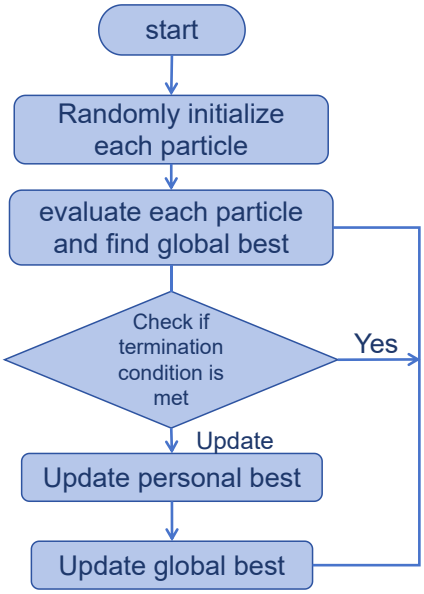


Figure 13. Flowchart of the PSO-based trajectory planning process.

3.4.2. Dataset Construction and Sample Overview

To train our sequence model for robotic arm trajectory prediction, we first generated a diverse set of joint-space trajectories using PSO-optimized waypoints and cubic interpolation, ensuring both feasibility and smoothness in joint motion. Each trajectory is defined by a random start and end configuration within the arm's reachable workspace, followed by the optimization of intermediate joint angles to minimize end-effector endpoint error and overall acceleration magnitude. After interpolation to a fixed length of 50 time steps, we recorded the arm's Cartesian end-effector position (x, y, z) at each step. By concatenating 2500 such trajectories, we obtained a dataset of 125,000 rows.

Table 8 presents a small excerpt from trajectory traj_id = 0, sampled every five-time steps. Each row corresponds to a distinct time index ('step') along the motion, and the three columns 'x', 'y', and 'z' give the end-effector's Cartesian coordinates in meters, computed from the forward kinematics of the optimized joint angles. This sample illustrates the gradual progression of the end-effector through space, which the trained model will learn to approximate from joint-space inputs.

Table 8. Sample end-effector positions for traj_id = 0 (every 5 steps).

Step	x (m)	y (m)	z (m)
0	1.1398	0.7503	−0.2995
5	1.1807	0.3654	−0.8995
10	1.2154	0.2448	0.0098
15	1.2089	0.3056	0.1295
20	1.1834	0.3594	0.2423
25	1.1417	0.4191	0.3904
30	1.0906	0.4478	0.2896
35	1.0116	0.4649	0.5895
40	0.9067	0.4825	0.6887
45	0.7715	0.4848	0.7603
50	0.6923	0.4699	0.7874

3.5. Transformer-KAN Prediction

3.5.1. Transformer

The Transformer is a neural network architecture based entirely on attention mechanisms, widely used in sequence modeling tasks [47]. Unlike traditional Recurrent Neural Networks (RNNs) [48] or Long Short-Term Memory (LSTM) networks [49], the Transformer dispenses with recurrence and relies on parallelized self-attention to model global dependencies between input and output.

The model consists of two main components: the **encoder** and the **decoder**, each composed of a stack of identical layers. The encoder processes the input sequence into a continuous representation, while the decoder generates the output sequence step by step using the encoder's outputs and its own generated tokens.

Positional Encoding: Since the Transformer lacks any inherent notion of sequence order, positional encoding is added to the input embeddings to provide information about the relative or absolute position of tokens. The most commonly used positional encoding is based on sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right), \tag{22}$$

where pos is the position index, i is the dimension index, and d_{model} is the model dimension.

Scaled Dot-Product Attention: The core mechanism of the Transformer is the attention function, which computes a weighted representation of the input sequence. Given queries Q, keys K, and values V, attention is computed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{23}$$

where d_k is the dimensionality of the keys.

Multi-Head Attention: Instead of performing a single attention function, the Transformer uses multiple attention heads to jointly attend to information from different representation subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O, \tag{24}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{25}$$

where W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices learned during training.

Feed-Forward Network: Each layer in both the encoder and decoder contains a fully connected feed-forward network applied to each position separately and identically:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{26}$$

Overall Architecture: Figure 14 illustrates the complete architecture of the Transformer model. On top, the encoder consists of stacked layers containing multi-head attention and feed-forward sublayers, each with normalization steps (Add and Norm). The input embeddings are first processed with positional encoding. On the bottom, the decoder also includes similar layers but with masked multi-head attention to prevent the decoder from attending to future tokens during training. The decoder also includes an encoder–decoder attention layer to help generate the output sequence based on the encoder's representation. Finally, the output is passed through a linear layer followed by a softmax layer to generate the output probabilities token by token.

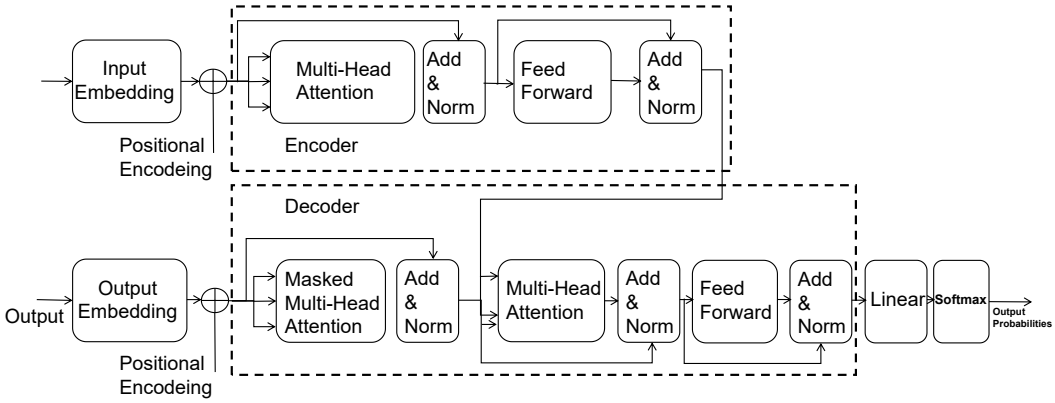


Figure 14. Transformer structure diagram.

3.5.2. Attention Mechanism: Kolmogorov–Arnold Networks (KAN)

A Multilayer Perceptron (MLP) serves as the fundamental and widely used deep learning model for approximating nonlinear functions. Its power in representation stems from the Universal Approximation Theorem, which states that an MLP can represent any continuous function. However, MLP does have significant limitations. In architectures such as transformers, MLP requires a large number of parameters, leading to challenges in model interpretability.

In contrast, Kolmogorov–Arnold Networks (KANs) introduce a different approach to function approximation. While MLP relies on nonlinear activation functions located within neurons, KAN applies nonlinear functions directly on the edges, or weights, between layers. These functions are learned and parameterized as splines, which enhances model interpretability and reduces the number of parameters required. This approach addresses some of the shortcomings of MLP.

To improve the performance of KAN, MIT introduced a method combining splines with MLP-like structures. Specifically, KAN integrates splines into the network, as shown in the comparative diagram (Figure 15).

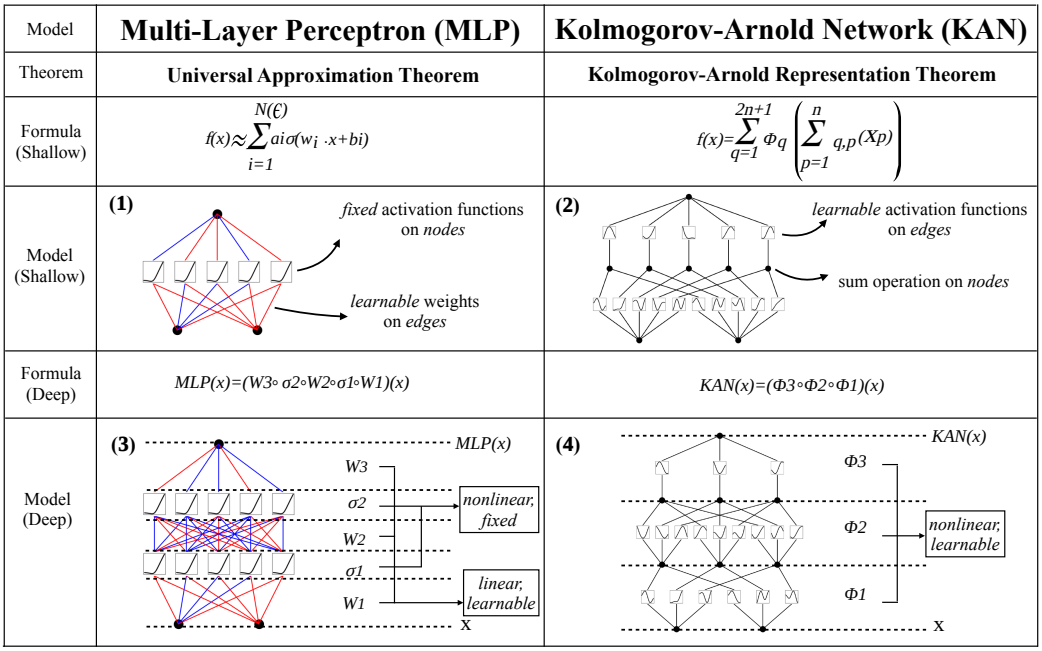


Figure 15. MLP vs. KAN [50].

The Kolmogorov–Arnold Representation Theorem plays a central role in KAN. The theorem asserts that any multivariate continuous function can be represented as a sum of univariate functions. KAN leverages this concept to represent complex data relationships with fewer parameters compared to traditional MLP, improving both computational efficiency and interpretability.

A KAN layer involves a transformation that combines compositions and additions of univariate functions. The basic structure of a KAN layer is as follows:

- $\mathbf{x^{(l)}}$: Input vector of layer l.
- $\mathbf{x^{(l+1)}}$: Output vector of layer $l+1$.
- $\phi_{l,i,j}$: Activation function connecting the i-th node of layer l to the j-th node of layer $l+1$.
- n_l: Number of nodes in layer l.
- n_{l+1}: Number of nodes in layer $l+1$.

The transformation from the input vector $x^{(l)}$ to the output vector $x^{(l+1)}$ in a KAN layer is performed through the following equation:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,i,j}(x_{l,i}) \quad \text{for} \quad j = 1, 2, \ldots, n_{l+1}, \tag{27}$$

where $x_{l,i}$ represents the input to the i-th node in layer l, and $\phi_{l,i,j}$ denotes the activation function applied to the weight connecting the i-th node in layer l to the j-th node in layer $l+1$.

The primary advantage of KAN is its ability to mitigate the vanishing gradient problem, which often arises with traditional activation functions like ReLU. By utilizing B-splines as the activation function, KAN can provide smoother transitions between layers, which results in improved gradient flow during training.

KAN also reduces the number of parameters compared to MLP, allowing them to learn complex patterns without the need for large networks. The use of edge-based nonlinear activation functions, rather than neuron-based activation functions, also simplifies the model structure and enhances interpretability.

The function $\phi(x)$, representing the activation function in KAN, is given by

$$\phi(x) = w(b(x) + \text{spline}(x)), \tag{28}$$

where $b(x)$ is a nonlinear activation function, specifically the SiLU (Sigmoid Linear Unit) function, defined as

$$\text{silu}(x) = \frac{x}{1 + e^{-x}}. \tag{29}$$

which has been found to work well in deep learning models.

A KAN consists of multiple layers, each with a transformation matrix that applies the nonlinear activation functions to the data. The full transformation process is described as follows:

1. First layer: The input vector $x^{(0)} = x$ is transformed into the output $x^{(1)}$ through the activation function matrix Φ_0.

$$x^{(1)} = \sum_{i=1}^{n_0} \phi_{0,i,j}(x_0^{(i)}). \tag{30}$$

2. Intermediate Layers: In each intermediate layer l, the output $x^{(l)}$ is computed through the transformation matrix Φ_l:

$$x^{(l+1)} = \sum_{i=1}^{n_l} \phi_{l,i,j}(x_l^{(i)}). \tag{31}$$

3. Final Layer: The final layer produces the output y:

$$y = \sum_{i=1}^{n_L} \phi_{L-1,i,j}(x_{L-1}^{(i)}). \tag{32}$$

KAN provides a compelling alternative to traditional MLP by replacing neuron-based activation functions with activation functions on the edges (weights). This leads to more parameter-efficient models with enhanced interpretability and the ability to approximate complex functions. The use of splines and other edge-based activations makes KAN a powerful tool in addressing the limitations of MLP and improving model performance across a wide range of deep learning tasks.

3.6. Transformer+KAN Model

Transformer+KAN integrates the robust capabilities of Transformer architectures with the powerful representation of Kolmogorov–Arnold networks (KANs) for improved efficiency and interpretability. In our framework, this hybrid model is used to predict the 3D spatial coordinates of the manipulator's end-effector, enabling the effective compensation of its planned trajectory. The Transformer model leverages self-attention mechanisms, and KAN replace traditional MLP layers with spline-based activation functions, enhancing expressiveness while reducing the number of parameters required [51]. The model structure follows a modular design, composed of several key components:

Input Layer: The input layer receives sequences of spatial pose trajectory coordinates. This could be in the form of a sequence of joint positions and orientations, crucial for the model to understand and predict the movements of an object, such as a robotic arm.
Transformer Layer: The Transformer layer processes the sequence data through multiple layers of Multi-Head Attention. Each attention sub-layer is followed by normalization and a residual connection to preserve input context and ease training.
KAN Layer: Unlike traditional transformers, which use MLP blocks, our model replaces them with KAN blocks. KAN uses spline-based learnable activation functions, providing richer nonlinear transformation capacity while maintaining better interpretability.
Attention Mechanism: Self-attention modules are used to model long-range dependencies in the input sequence. These are essential for encoding relationships across time or space, depending on the domain.

3.7. Visual Servo Simulation Based on MATLAB

In this chapter, we present a comprehensive overview of a vision-driven robotic arm control system designed to execute complex tasks with high precision. The system combines advanced image analysis and deep learning techniques, enabling dynamic interaction with its environment. At the core lies a visual servoing framework, which guides the robotic arm's movements by interpreting visual data captured from its surroundings and aligning its actions with predefined operational objectives.

As illustrated in Figure 16, the system begins by acquiring images through vision sensors, which are then processed using the YOLOv8 model for object detection. The extracted object features are compared against reference features corresponding to target

objects defined within the system. Discrepancies between the detected and target features are computed, serving as critical inputs to the arm's joint-level controller.

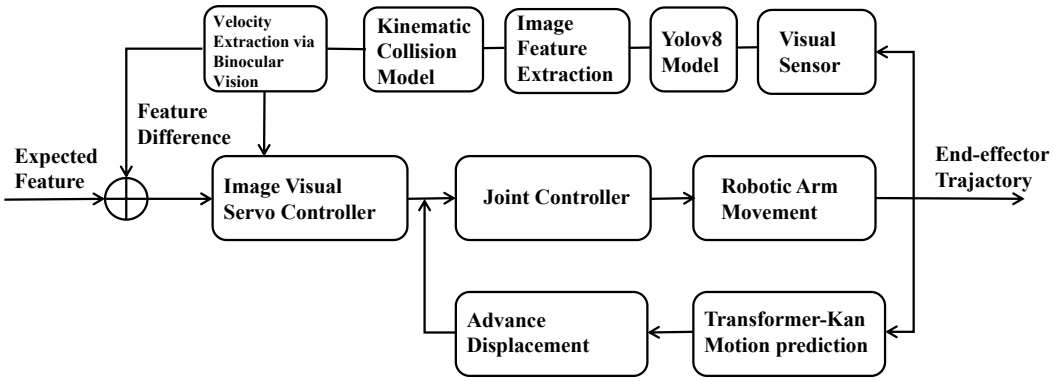


Figure 16. Visual servo simulation flowchart.

This controller utilizes the feature differences, along with the Jacobian matrix, to estimate the optimal joint angles across all six degrees of freedom. Based on these calculations, it generates movement commands that incrementally adjust the robotic arm's posture to converge toward the intended configuration.

To further enhance tracking and responsiveness, the system incorporates a Transformer-KAN motion prediction module. This component leverages the historical trajectory of the arm's end-effector to forecast future positional shifts. The predicted positions are translated into joint angle estimations, which are then integrated into the control logic. By incorporating these predictive adjustments, the robotic arm is able to anticipate target motion trends, allowing for smoother and more effective tracking performance.

4. Results and Discussion

4.1. Target Detection

In this section, we evaluate the object detection performance of our model using standard metrics, including Precision, Recall, Average Precision (AP), mean Average Precision (mAP), and the F_1 score. These metrics collectively offer a comprehensive assessment of the model's ability to detect and classify the target class—debris—against a background.

Precision and recall are calculated by ranking the detection outputs according to their confidence scores. Precision measures the proportion of true positive detections among all predicted positives, while recall measures the proportion of true positives among all actual objects. Their definitions are given by

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \tag{33}$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively.

The F_1 score, which balances both precision and recall, is computed as

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{34}$$

As shown in Figure 17, the precision–recall curve for the "debris" class remains consistently high across a wide range of recall values, achieving an average precision (AP)

of 0.955. The model's mean average precision (mAP) at an IoU threshold of 0.5 is also calculated. mAP represents the mean value of AP over all classes and is defined as

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{35}$$

where N is the number of classes, and AP_i is the average precision of class i.

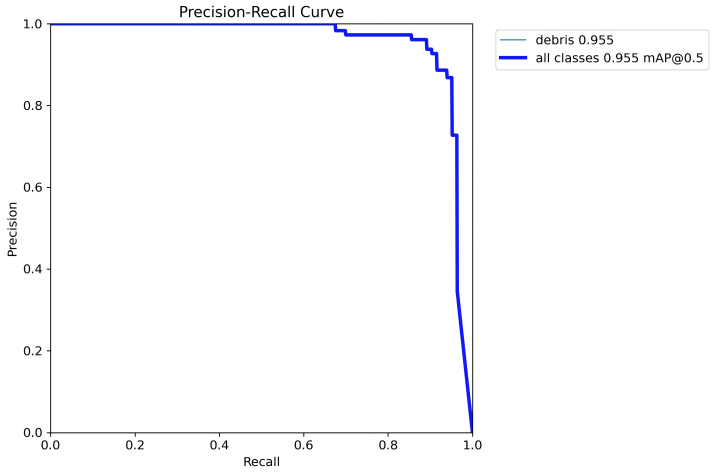


Figure 17. Precision–recall curve for the "debris" class.

To further explore the influence of confidence thresholds, we examine the relationships between confidence and performance metrics. In Figure 18, the F_1 score peaks at 0.92 at a threshold of 0.716, reflecting the optimal balance point. In Figure 19, precision reaches 1.00 when the threshold is increased to 0.961, and Figure 20 confirms that recall stays near-perfect at lower thresholds.

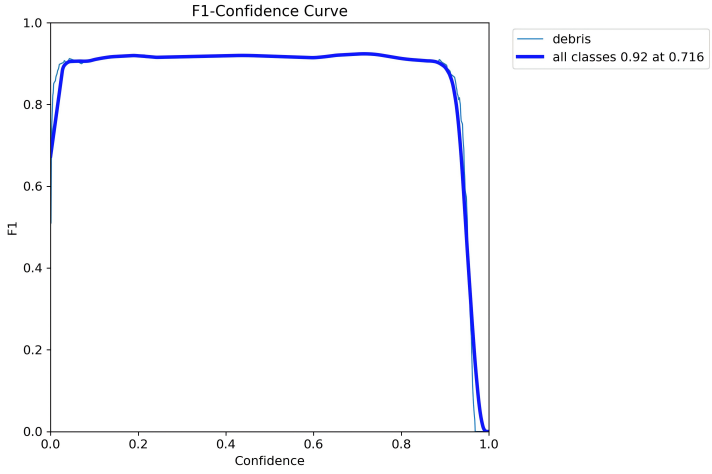


Figure 18. F_1 score vs. confidence threshold.

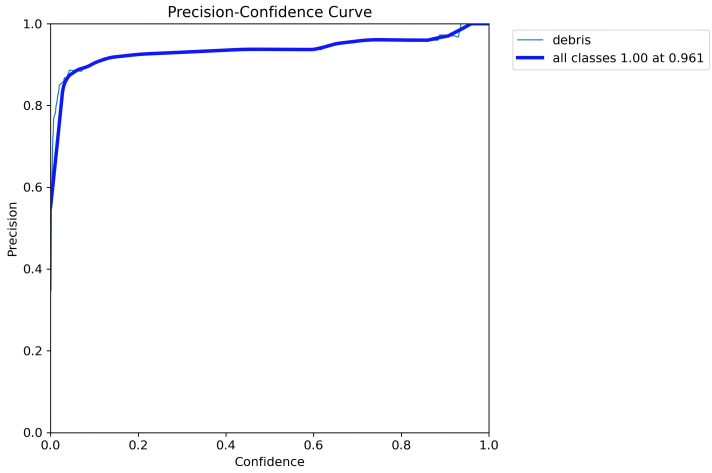


Figure 19. Precision vs. confidence threshold.

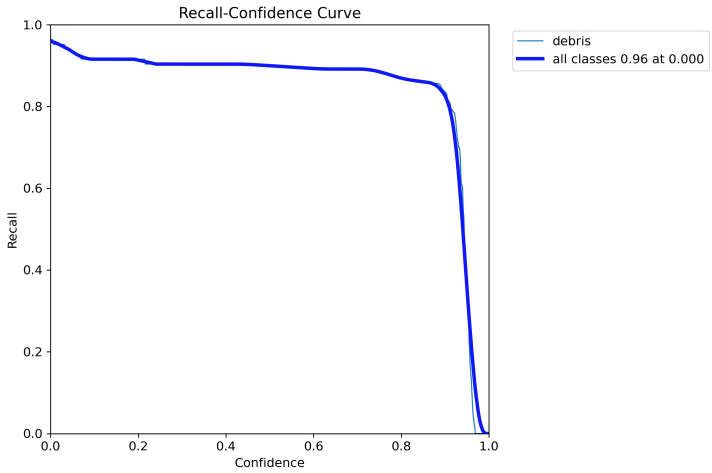


Figure 20. Recall vs. confidence threshold.

The model's training progress is visualized in Figure 21. The box loss, classification loss, and distribution focal loss (DFL) all decrease steadily during training. Evaluation metrics such as mAP and F_1 also improve continuously, reflecting stable convergence and good generalization ability.

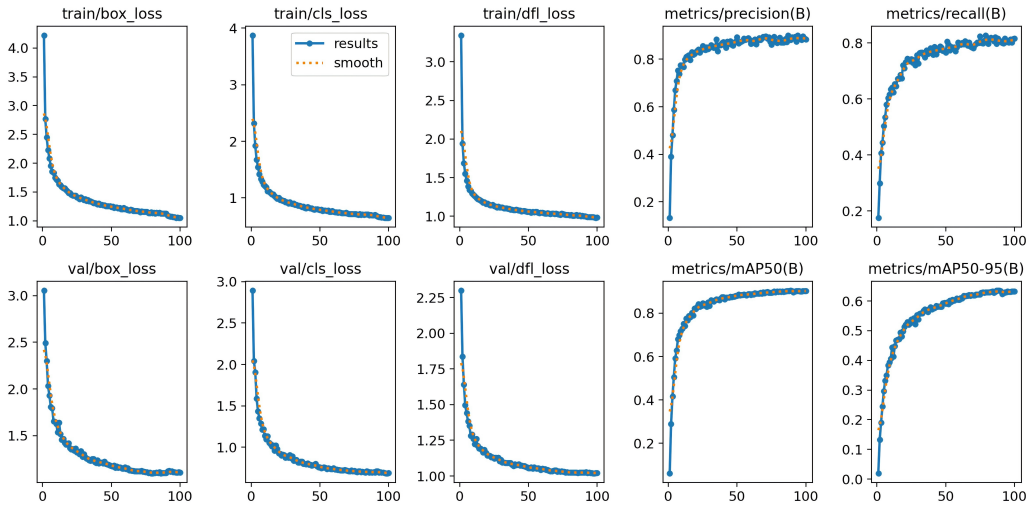


Figure 21. Training and validation metrics across 100 epochs.

We also visualize the spatial distribution and size of detected objects to assess the model's localization behavior. As shown in Figure 22, debris instances are well-distributed across the image, and the bounding box sizes remain consistent, indicating that the model avoids positional bias and detects targets with stable scale inference.

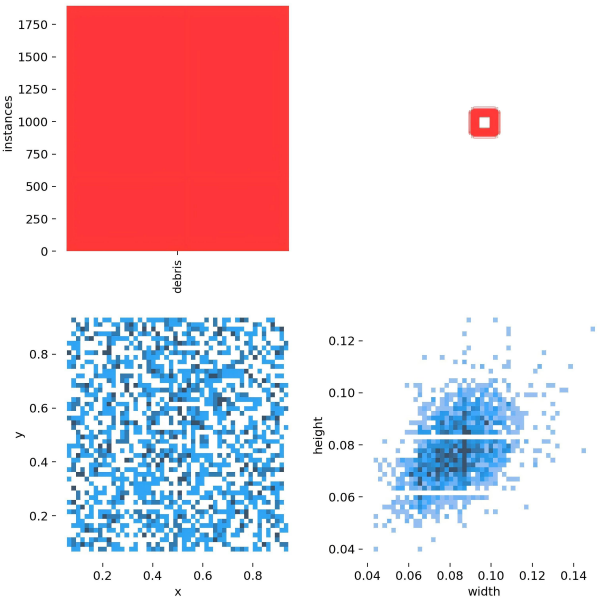
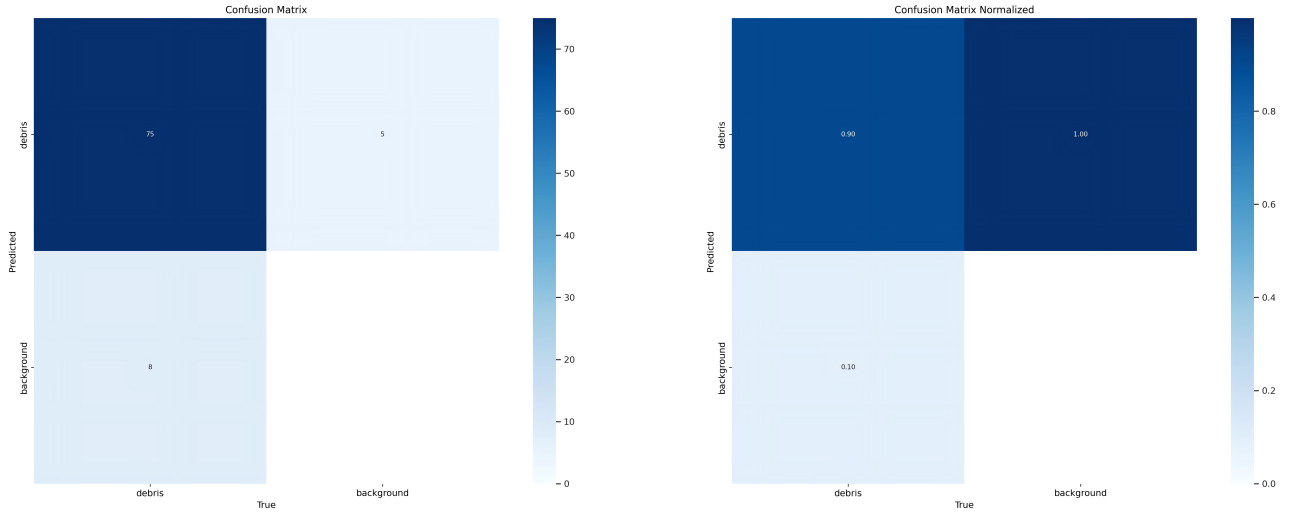


Figure 22. Heatmap and box size distribution of debris detections.

To evaluate classification performance, we analyze the confusion matrices in Figure 23. Figure 23a shows the raw confusion matrix, where the model correctly classifies nearly all debris instances with minimal confusion. Figure 23b, the normalized version, further confirms strong class separation, with nearly all predictions lying along the diagonal.



(**a**) Confusion matrix for validation data. (**b**) Normalized confusion matrix.

Figure 23. Comparison of raw and normalized confusion matrices.

To support our model selection, we conducted a quantitative comparison among DETR, RT-DETR, YOLOv5, and YOLOv8 using standard evaluation metrics, including Precision (P), Recall (R), mAP@50, and mAP@50:95. As shown in Figure 24, YOLOv8 consistently outperforms the other models across all evaluation metrics. Specifically, it achieves the highest precision (93.5%), recall (86.5%), mAP@50 (95.5%), and mAP@50:95

(66.9%), which clearly demonstrates its superior ability to detect and localize the target class with both accuracy and generalization. Moreover, despite its outstanding performance, YOLOv8 maintains a remarkably small parameter size (3.01M) and low computational complexity (8.3 GFLOPs), making it particularly suitable for deployment in real-time or resource-constrained environments.

Therefore, considering both detection performance and model efficiency, YOLOv8 was chosen as the primary detection backbone in this work. Its superior balance of accuracy and speed ensures that it not only delivers state-of-the-art results but also scales well for practical application scenarios.

To assess the real-time performance of the final detection model, we conducted a benchmark test using a standard YOLOv8 model deployed on an NVIDIA RTX 4060 GPU, as shown in Table 9. The input resolution was set to 640 × 640, and the batch size was fixed at 4. A total of 500 test images were processed using the official Ultralytics YOLOv8 framework. The total inference time was measured at 2.94 s, resulting in an average inference speed of approximately **170 FPS**. This demonstrates that, despite the incorporation of additional modules such as stereo vision and trajectory estimation, the system remains computationally efficient and suitable for real-time applications in space debris detection.

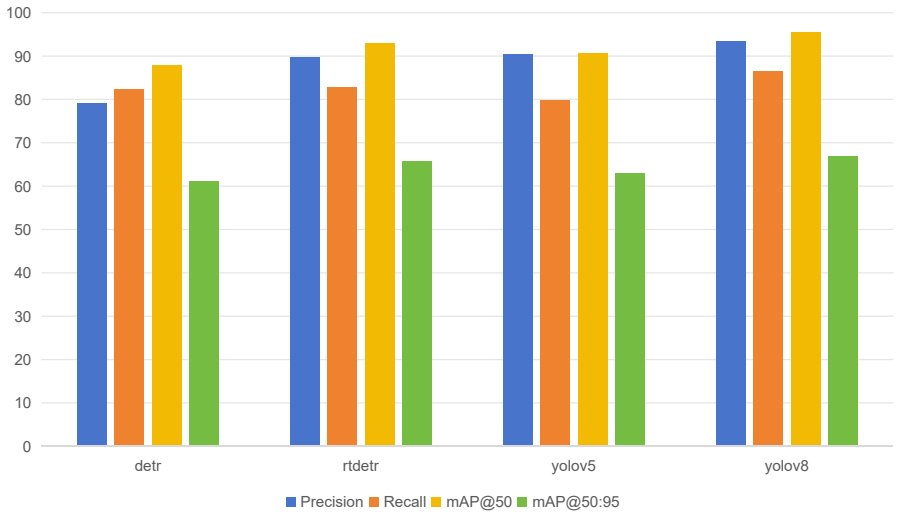


Figure 24. Comparative bar chart showing precision, recall, mAP@50, and mAP@50:95 across different detection models. YOLOv8 outperforms other models in all metrics.

Table 9. Comparison of detection models on the same dataset.

Model	Epochs	Batch Size	Learning Rate	mAP@0.5 (%)
DETR [18]	100	2	1×10^{-4} (cosine)	87.95
RT-DETR [52]	100	4	1.91×10^{-3}	93.00
YOLOv5 [36]	100	8	1.85×10^{-3}	90.70
YOLOv8	**100**	**4**	$\mathbf{1.94 \times 10^{-3}}$	**95.50**

Finally, to realistically simulate space debris in a controlled visual environment, we conducted a simplified modeling process using MATLAB R2024a. In this simulation, two white spherical objects of approximately the same size were used to represent space debris. The spheres were placed in a dark scene with a black background to mimic the low-illumination conditions of outer space. A single point light source was introduced and its angle was adjusted to cast realistic highlights and shadows on the surfaces of the spheres, enhancing contrast and replicating the optical features of space-based imagery.

The reason for selecting debris targets of similar size in this experiment is two-fold. First, it allows the object detection model to rapidly localize each target and treat them

as individual points in space, simplifying the visual inference process. Second, this setup enables the more effective extraction of spatial and kinematic parameters—such as velocity vectors, movement direction, and trajectory prediction—for the subsequent analysis of potential collisions. By using geometrically uniform shapes, the focus of the experiment remains on evaluating the detection performance and motion estimation under space-like visual conditions, rather than introducing additional complexity due to scale variation. Figure 25 shows an example detection result on a test image. The model accurately localizes multiple debris targets, each annotated with confidence scores above 0.69. This visual outcome reinforces the strong quantitative metrics reported above and illustrates the model's effectiveness in real-world inference.

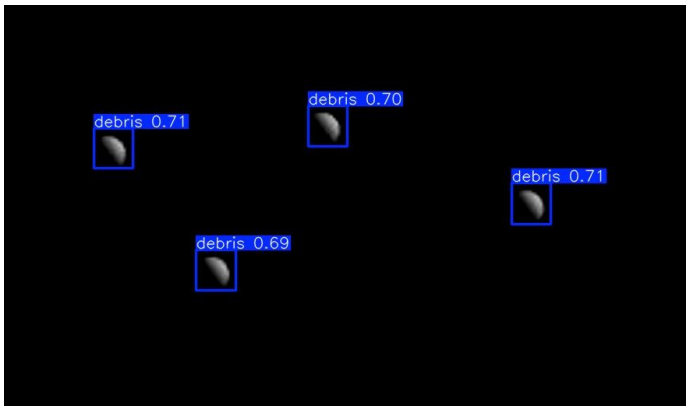


Figure 25. Detection result on test image showing multiple correctly localized debris objects with confidence scores.

The modeling and simulation process was conducted in MATLAB. Two cameras were positioned in the 3D space for this simulation. Camera 1 was placed at the coordinates (2.0, −1.0, 1.5) and oriented to face the point (1, 1, 1), while Camera 2 was placed at (0.3, 2.2, 1.4), also facing towards the same point (1, 1, 1). The detection views from both cameras will be shown in Figure 26.

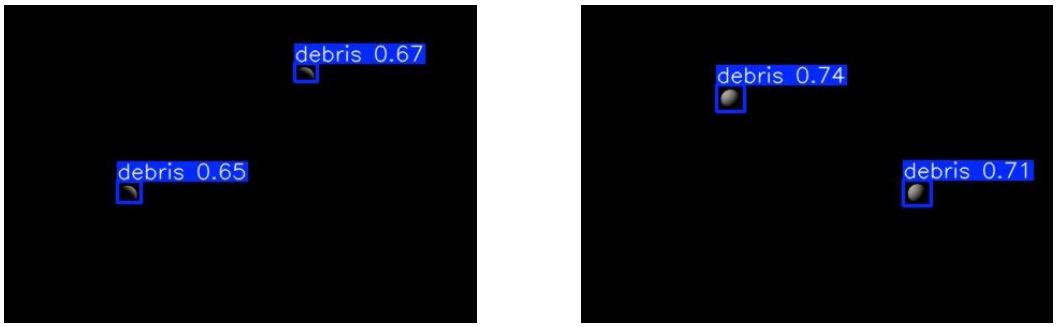


Figure 26. Left: View from Camera 1 positioned at (2.0, −1.0, 1.5). **Right**: View from Camera 2 positioned at (0.3, 2.2, 1.4). Both cameras are observing two simulated debris objects in 3D space, oriented toward the same target point (1, 1, 1).

Based on the velocity extraction via binocular vision described in Section 3.3.1 and the kinematic model of spatial binary-sphere collision outlined in Section 3.3.2, we calculated the velocities of two spheres in 3D space that are likely to collide. Specifically,

Sphere 1 has an initial velocity of [0.3, 0.3, 0.3], moving along the spatial diagonal;

Sphere 2 has an initial velocity of [−0.3, −0.3, −0.29], in an approximately opposite direction.

According to the kinematic model, the two spheres are expected to collide at time t = 1.1667 s at the spatial coordinate [0.55, 0.55, 0.65]. At t = 1.2 s, the positions of the two spheres are observed to be [0.531, 0.531, 0.684] and [0.569, 0.569, 0.528], respectively. The robotic manipulator's end-effector will subsequently perform servo control targeting Sphere 1's predicted position at that moment. Using the conservation of momentum and the restitution coefficient (e = 0.9).

We then simulated the entire collision process in MATLAB, producing a 3D visualization that shows the trajectories of both spheres before and after the collision, as illustrated in Figure 27.

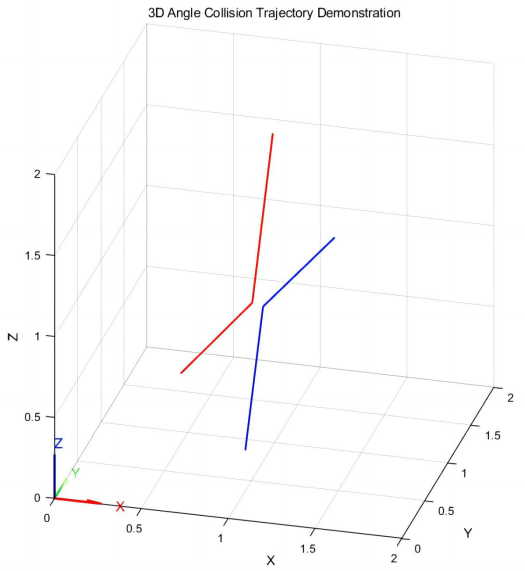


Figure 27. Trajectory Prediction of Binary-Sphere Collision, where the blue and red lines represent different ball movement trajectories.

4.2. Robotic Arm Trajectory Planning

Based on the robotic arm parameters presented in Table 10, the robotic arm is modeled in MATLAB, as shown in Figure 28.

The following figures illustrate the impact of employing the PSO algorithm to enhance the trajectory planning of robotic joints. As depicted in Figure 29, the evolution of each joint's historical best fitness value is tracked across optimization iterations, highlighting the algorithm's convergence behavior. A detailed comparison of joint position, velocity, and acceleration profiles—before and after optimization—offers clear insight into the improvements achieved. Figure 30 presents this comparative analysis, showcasing the effectiveness of the PSO-based optimization approach.

Table 10. DH parameters for the robotic arm.

Link	d (m)	a (m)	α (rad)	Offset (rad)
1	0.5	0.2	$-\frac{\pi}{2}$	0
2	0	0.79	0	$-\frac{\pi}{2}$
3	0	0.14	$-\frac{\pi}{2}$	0
4	0.78	0	$\frac{\pi}{2}$	0
5	0	0	$-\frac{\pi}{2}$	0
6	0.104	0	0	0

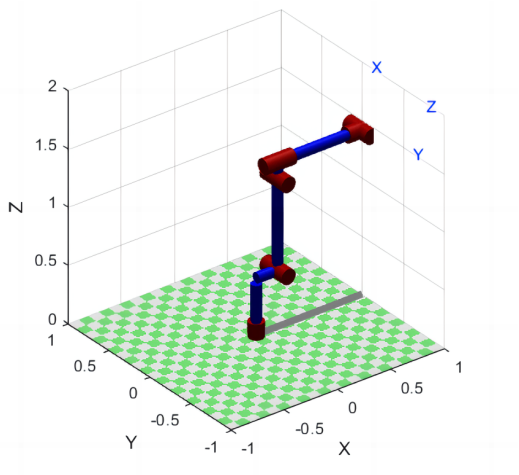


Figure 28. Modeling the robotic arm in MATLAB.

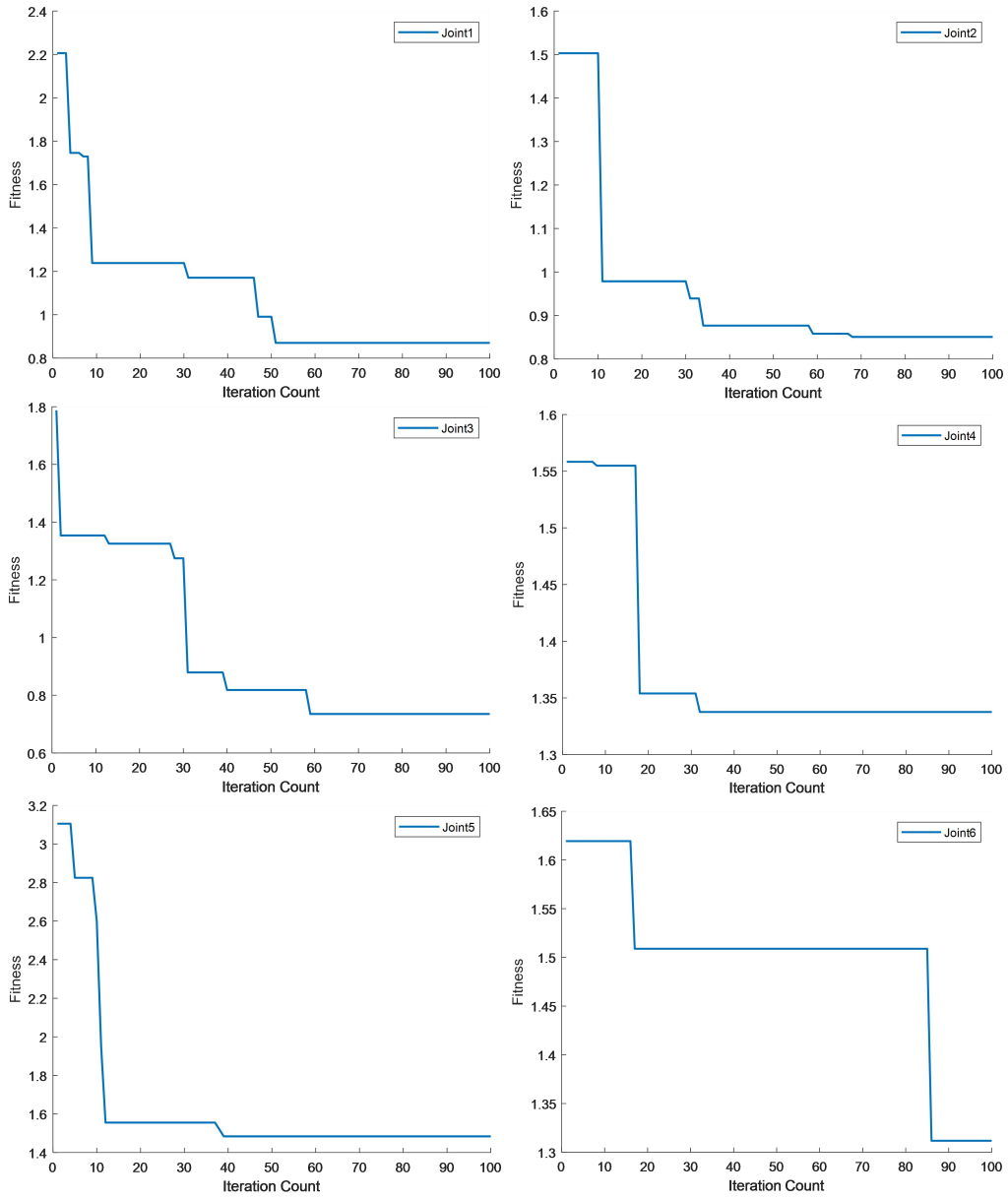


Figure 29. Joint kinematic comparison before and after PSO-based trajectory optimization. The **left** column shows the non-optimized position, velocity, and acceleration profiles, while the **right** column presents the corresponding optimized curves.

Comparing the unoptimized and optimized joint angle trajectories of the 6-DOF robotic arm shows clear performance gains. In the unoptimized case (left column), the motion takes about 6 s, with joint angles sweeping roughly -1.0 rad to $+2.2$ rad. Although each curve is smooth, poor synchronization causes uneven timing and inefficiencies. After optimization (right column), the same maneuver completes in about 2.2 s, with all peak displacements constrained to approximately ±3 rad, yielding much tighter coordination and reduced overshoot.

Differences in joint speed profiles are even more pronounced. Unoptimized joint velocities peak near ± 2 rad/s and display slow, staggered transitions that lead to lag across axes. In the optimized case, peak speeds climb to nearly ± 2.5 rad/s, and the velocity peaks across all joints become closely aligned. Despite the higher speeds, the curves remain smooth—evidencing a balanced plan that boosts throughput without sacrificing continuity.

Acceleration profiles highlight system responsiveness. The original acceleration curves stay within ± 5 rad/s^2, reflecting a conservative, cautious controller. Post-optimization, accelerations reach up to ± 15 rad/s^2 yet avoid sharp spikes, delivering noticeably greater responsiveness, improved path adherence, and ultimately shorter cycle times with less mechanical wear.

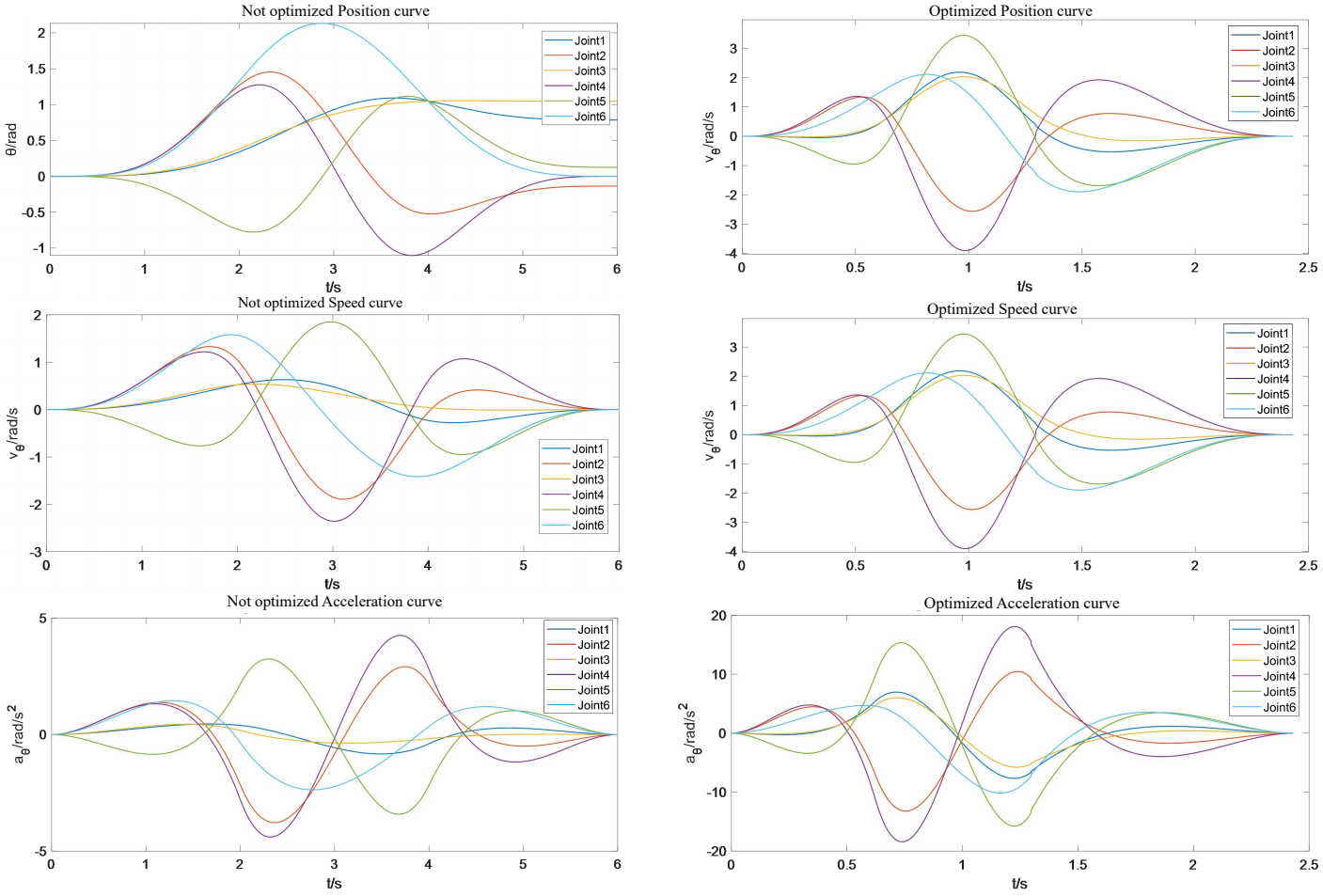


Figure 30. Joint kinematic comparison before and after PSO-based trajectory optimization. The **left** column shows non-optimized position, velocity, and acceleration profiles, while the **right** column presents the corresponding optimized curves.

4.3. Transformer-KAN Prediction

To assess the impact of introducing Kernel-based Attention Networks (KANs) into the Transformer architecture, we conducted a series of controlled experiments. The evaluation

leverages three common regression metrics—Huber Loss, root mean squared error (RMSE), and coefficient of determination (R^2)—as well as trajectory prediction comparisons. Below, we present and analyze the corresponding results in detail.

We begin by introducing the definitions of the evaluation metrics:

Huber Loss:

$$L_\delta(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & |y_i - \hat{y}_i| \le \delta, \\ \delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right), & |y_i - \hat{y}_i| > \delta\,. \end{cases} \tag{36}$$

Huber loss combines the sensitivity of squared error for small residuals with the robustness of absolute error for larger deviations, thereby reducing the influence of outliers.

Root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{37}$$

RMSE shares the same units as the original target variable, providing a more interpretable measure of absolute prediction error.

Coefficient of determination (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{38}$$

where $\bar{y}$ is the mean of the actual values. R^2 reflects how much variance in the true values is captured by the model; values close to 1 indicate an excellent fit, while values below 0 imply that the model performs worse than a constant predictor.

Training loss (Huber) analysis: Figure 31 plots the Huber loss over 300 training epochs for the Transformer model. The curve begins at a moderate initial value of around 0.06, reflecting the model's early prediction errors under the robust Huber metric. It then drops precipitously within the first 5–10 epochs—from about 0.06 down to roughly 0.012—indicating very rapid initial learning.

Following this burst of improvement, the loss curve oscillates modestly and enters a slower, steady descent. From approximately epoch 10 to 50, the Huber loss levels off near 0.009, fluctuating only slightly as the model fine-tunes. Beyond epoch 50, the loss continues to decrease gradually, reaching about 0.004 by epoch 200.

In the late stage of training (epochs 200–300), both the training and test curves converge around 0.002, signifying that further epochs yield only marginal gains. The tight overlap between the train and test losses throughout also suggests strong generalization with minimal overfitting.

In contrast, Figure 32 presents the Huber loss of the Transformer+KAN model over 270 training epochs. The curve begins at approximately 0.0098, then falls sharply to about 0.005 by epoch 20, indicating rapid initial learning. Between epochs 20 and 50, the loss continues a smooth decline to around 0.003, after which it enters a steadier, slower descent, reaching roughly 0.002 by epoch 100. In the late training stage (epochs 100–270), the loss tapers off and converges near 0.001 by epoch 270. Throughout training, the train and test curves nearly overlap, demonstrating strong generalization and minimal overfitting. This behavior shows that integrating KAN not only accelerates convergence but also yields a significantly lower final error, reflecting an enhanced capacity to model complex nonlinear and temporal dependencies.

Throughout training, the test loss (orange) nearly overlaps the train loss (blue), with only minor fluctuations, indicating excellent generalization and minimal overfitting. This behavior shows that adding KAN not only accelerates early learning but also drives the model to a significantly lower final error, underscoring its enhanced ability to capture complex nonlinear and temporal patterns.

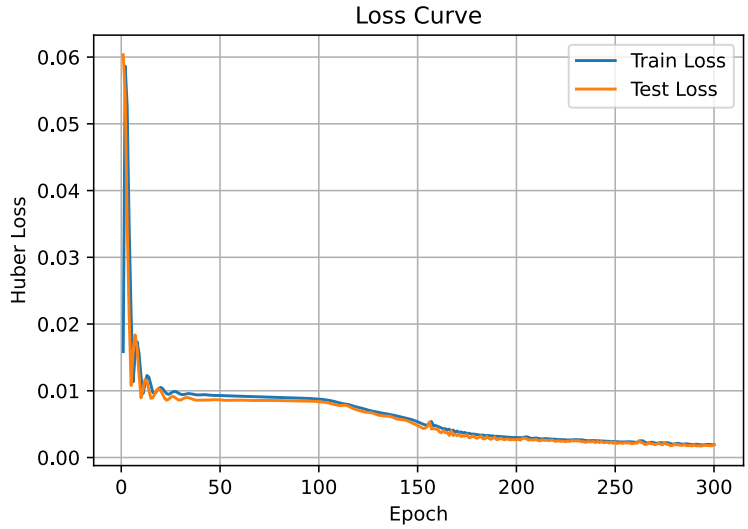


Figure 31. Huber loss curve for the Transformer model.

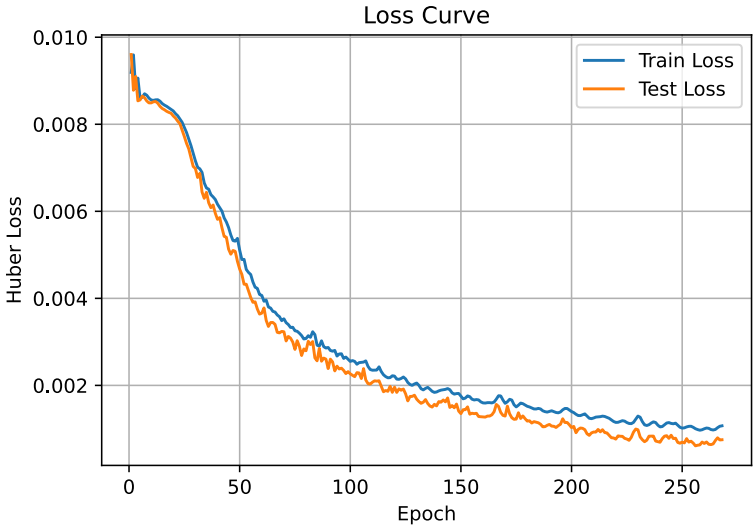


Figure 32. Huber loss curve for the Transformer+KAN model.

RMSE Comparison: As shown in Figure 33, it plots the RMSE of the Transformer model over 300 training epochs. The RMSE decreases from an initial value of roughly 5.0 m at epoch 0 to about 1.0 m by epoch 1, demonstrating a steep initial learning. Over the next 20 epochs, the error falls to approximately 0.8 m, followed by a smoother decline to 0.6 m at epoch 100 and 0.4 m at epoch 150. In the late stage (epochs 150–300), RMSE continues to improve gradually, reaching 0.3 m at epoch 200, 0.2 m at epoch 250, and approaching 0.1 m by epoch 300. The near-coincidence of training and test curves throughout indicates strong generalization and minimal overfitting. These trends suggest that the Transformer quickly captures dominant low-frequency patterns, while the further refinement of higher-frequency components yields diminishing returns over extended training.

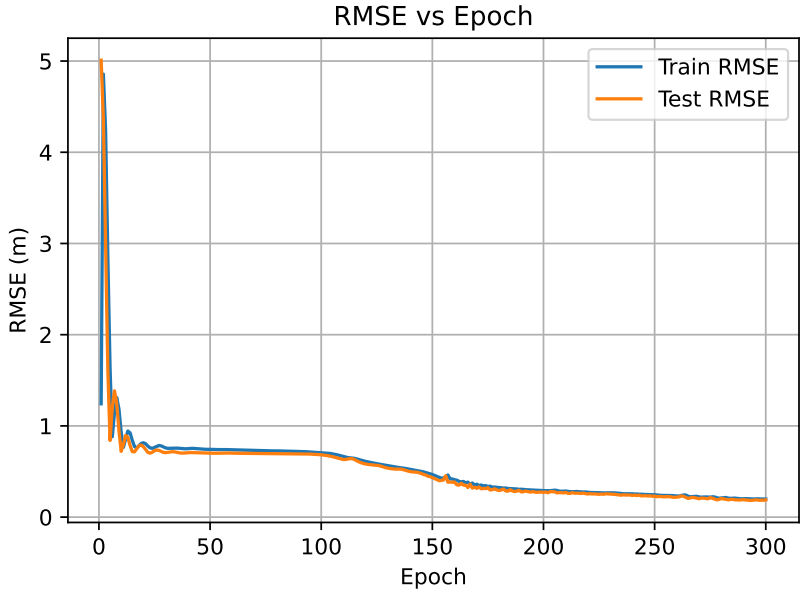


Figure 33. RMSE during training for Transformer.

In contrast, Figure 34 demonstrates that Transformer+KAN maintains a steady downward trend throughout training, ultimately reaching an RMSE of approximately 0.12. The absence of strong fluctuations and the continuous decline suggest a more stable and effective learning process. The reduced RMSE indicates that the KAN module helps the model minimize absolute prediction errors more consistently across epochs.

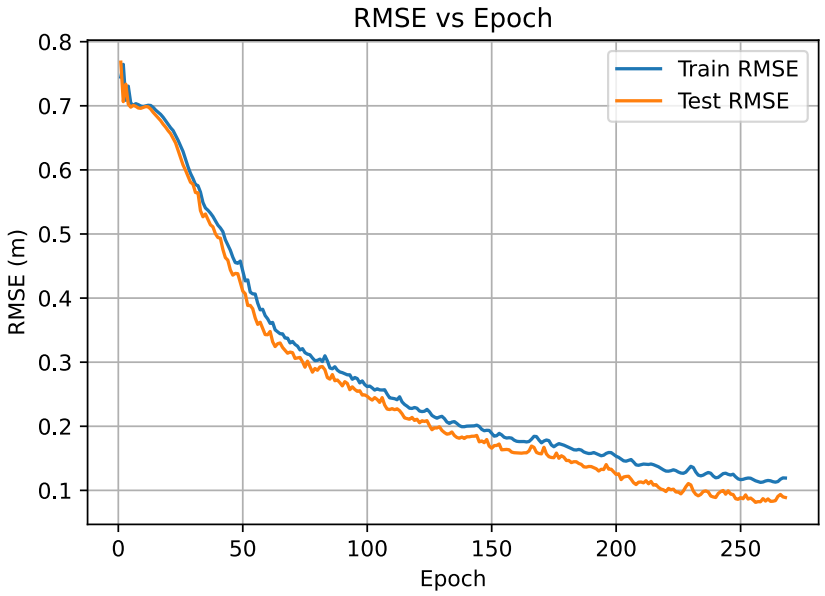


Figure 34. RMSE during training for Transformer+KAN.

R^2 score evolution: Figure 35 plots the R^2 progression for the Transformer model. Initially, the model exhibits highly unstable R^2 values below 0.5, with noticeable noise in the first 100 epochs. It eventually stabilizes, but only reaches around 0.82 by the end of training. This indicates that roughly 18% of the variance in the data remains unexplained.

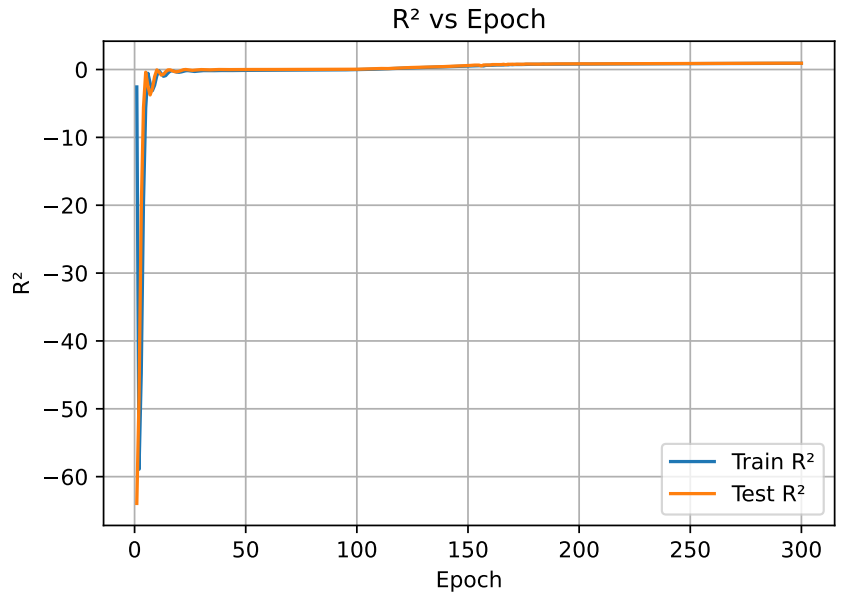


Figure 35. R^2 score progression for the Transformer.

In comparison, the Transformer+KAN model shown in Figure 36 follows a much more desirable trajectory. Although it also starts with erratic fluctuations, R^2 begins to rise steadily after epoch 80. By epoch 200, it surpasses 0.9 and ultimately reaches around 0.935. This high final score signifies that the model captures nearly all the variance in the data, showcasing its superior generalization ability.

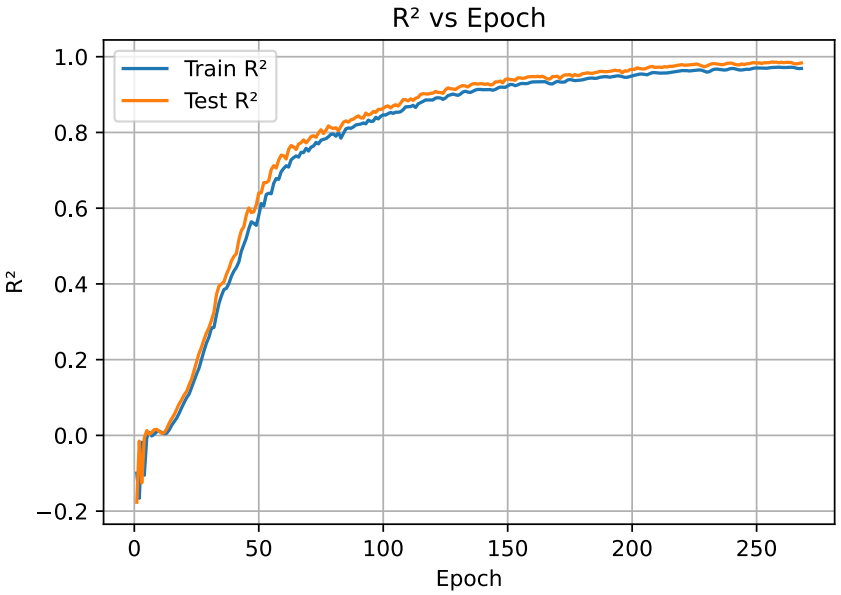


Figure 36. R^2 score progression for Transformer+KAN.

Trajectory prediction on X axis: The effectiveness of the model in capturing real-world temporal dynamics is further demonstrated by the predicted trajectory in the X-axis dimension, as shown in Figure 37. The Transformer+KAN model generates predictions that closely align with the ground truth across the entire time span. It successfully reproduces the amplitude and frequency of oscillations, including sharp peaks and valleys, with high fidelity.

In contrast to the baseline Transformer, the +KAN architecture displays a significantly better ability to represent high-frequency components and non-stationary transitions. This

advantage can be attributed to the KAN module's capacity to capture localized dynamic behavior, making it particularly suitable for complex motion forecasting tasks.

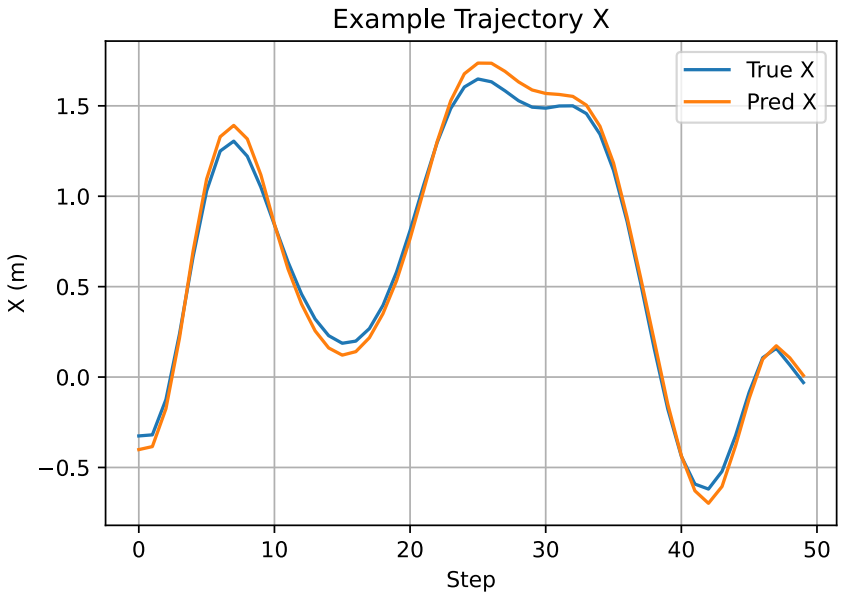


Figure 37. Trajectory prediction on the X axis using Transformer+KAN.

In conclusion, the Transformer+KAN architecture consistently outperforms the vanilla Transformer across all evaluation metrics. The combination of faster convergence, lower prediction error, higher explanatory power, and stronger trajectory fidelity confirms that the integration of KAN significantly enhances the model's ability to learn complex temporal dependencies in sequential data.

5. Conclusions

5.1. Framework Overview and Validation

Although our system integrates multiple functional modules—including space debris detection, trajectory prediction, and robotic arm motion planning—each module has been deliberately selected and implemented to strike a balance between accuracy, speed, and computational efficiency. Rather than relying on overly heavy or redundant architectures, we emphasize modularity and lightweight design principles throughout the pipeline.

The YOLOv8 model, used for space debris detection, is well-known for its lightweight architecture and fast inference speed, making it particularly suitable for real-time applications where resource constraints are common. In our implementation, it operates at a reduced resolution (256 × 256), which helps minimize computational overhead while still preserving detection performance.

For trajectory prediction, the Transformer+KAN architecture is used to boost prediction accuracy and convergence speed without incurring significant extra computational cost. The KAN module introduces parameter sparsity and better interpolation behavior, which helps reduce the overall training time and improves trajectory fidelity.

Moreover, the system is implemented in a modular fashion, allowing each component to run independently or be replaced without re-engineering the entire pipeline. This modularity supports future hardware deployment and real-world adaptability. The lightweight architecture, GPU-level optimization, and bounded memory usage together ensure that the overall system remains computationally feasible and suitable for embedded or on-board applications in space robotics.

5.2. Further Work

While the system has demonstrated promising results in controlled simulations, there are several areas that require targeted improvement and further expansion to enhance its robustness, versatility, and real-world deployment readiness.

First, our dataset covers primarily meteoroid-like debris, overlooking satellite fragments, spent rocket stages, and other objects of varied size, shape, and composition. Gathering additional imagery and sensor data to include these diverse debris types will strengthen detection robustness and recognition generalization, enabling the model to maintain high accuracy across a broader range of orbital debris scenarios.

Second, experiments have so far been conducted as independent MATLAB simulations, offering initial functional insights but lacking integration with flight-like hardware. Migrating the entire workflow into the Robot Operating System (ROS) framework will create a unified simulation environment, combining perception, prediction, and motion modules. This integrated ROS testbed will validate algorithm–hardware compatibility, latency, and real-time performance under more realistic, dynamic conditions.

Third, the current trajectory prediction model does not account for either debris mass or physical dimensions, both of which significantly influence inertial properties and system responses to control inputs—especially in high-precision tasks such as collision avoidance and interception timing. Future work should aim to estimate both mass and size from visual or multispectral data. This could involve a YOLOv11-based volume inference module that not only approximates spatial dimensions but also infers mass through material density assumptions. Such a dual-parameter approach would enable mass- and size-aware motion predictions, offering a more faithful reflection of real-world dynamics under varying force and torque conditions.

In parallel, the motion model presently assumes straight-line, constant-velocity trajectories, which overlooks perturbations from Earth's non-uniform gravity field, atmospheric drag at higher altitudes, and interactions among debris fragments. Incorporating force-aware motion models that account for these perturbative effects will yield more accurate and realistic trajectory forecasts.

In addition, while our current trajectory control strategy primarily operates at the kinematic level, the inclusion of dynamic-level modeling and control is essential—particularly in microgravity environments where precision manipulation is sensitive to inertia, joint torques, and external disturbances. Future work will extend the control framework to incorporate robotic arm dynamics, enabling torque-aware planning, stability guarantees, and the more robust execution of capture and deflection maneuvers under realistic space conditions. To this end, we also plan to establish a laboratory-based space-simulation environment and implement the system using an Estun E20 robotic arm. This hardware-in-the-loop testbed will allow us to replicate aspects of microgravity and validate the integrated framework under physically realistic conditions, thereby enhancing the practical applicability of the proposed approach.

Exploring reinforcement learning or meta-learning could provide adaptive control strategies, enabling the manipulator to learn continually and adjust to changing conditions and unforeseen obstacles, thereby boosting robustness in unpredictable environments.

Real-time processing under spaceborne constraints remains challenging. Optimizing computational efficiency—through model pruning, quantization, or deployment on FPGAs/GPUs—will be crucial to ensure onboard performance within limited resources. In parallel, we plan to investigate model lightweighting techniques and modularized software design, with the goal of enabling scalable deployment on embedded hardware platforms. These efforts will simplify system integration, improve reproducibility, and enhance adaptability to real-world mission constraints.

Another important aspect concerns image degradation under realistic space conditions, such as haze, glare, or motion blur, which can impair detection accuracy. Future work will explore the integration of image enhancement and dehazing techniques, including contrast adjustment and restoration-based methods, to preprocess raw sensor inputs. This preprocessing step could mitigate visual distortions and provide clearer inputs for the detection model, thereby improving recognition reliability under degraded visual conditions.

In summary, while feasibility for autonomous debris removal has been demonstrated, advancing dataset diversity, simulation realism, dynamic modeling, and control, laboratory-based space-simulation testing, mass and force estimation, adaptive learning, model lightweighting, and embedded deployment, computational efficiency, and robust perception under degraded imagery is essential to achieve fully autonomous, operational space debris remediation.

Author Contributions: Conceptualization, Z.Z. and B.H.S.A.; methodology, Z.Z., D.Z. and B.H.S.A.; resources, B.H.S.A. and Z.Z.; writing—original draft preparation, B.H.S.A. and Z.Z.; writing—review and editing, B.H.S.A. and Z.Z.; supervision, B.H.S.A.; visualization, B.H.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Agency, E.S. ESA Space Environment Report. 2024. Available online: https://www.esa.int/Space_Safety/Space_Debris/ESA_Space_Environment_Report_2024 (accessed on 20 July 2025).
2. Mark, C.P.; Kamath, S. Review of active space debris removal methods. *Space Policy* **2019**, *47*, 194–206. [CrossRef]
3. Fallahiarezoodar, N.; Zhu, Z.H. Review of autonomous space robotic manipulators for on-orbit servicing and active debris removal. *Space Sci. Technol.* **2025**, *5*, 0291. [CrossRef]
4. Zhou, M.; Liu, T.; Wei, C.; Yu, Y.; Yin, J.; Wang, X. Improved Reinforcement Learning–Based Method for Emergency Mission Planning to Remove Space Debris. *J. Aerosp. Eng.* **2025**, *38*, 04025020. [CrossRef]
5. Kessler, D.J.; Cour-Palais, B.G. Collision frequency of artificial satellites: The creation of a debris belt. *J. Geophys. Res. Space Phys.* **1978**, *83*, 2637–2646. [CrossRef]
6. Forshaw, J.L.; Aglietti, G.S.; Navarathinam, N.; Kadhem, H.; Salmon, T.; Pisseloup, A.; Joffre, E.; Chabot, T.; Retat, I.; Axthelm, R.; et al. RemoveDEBRIS: An in-orbit active debris removal demonstration mission. *Acta Astronaut.* **2016**, *127*, 448–463. [CrossRef]
7. Flores-Abad, A.; Ma, O.; Pham, K.; Ulrich, S. A review of space robotics technologies for on-orbit servicing. *Prog. Aerosp. Sci.* **2014**, *68*, 1–26. [CrossRef]
8. Davis, J.P.; Mayberry, J.P.; Penn, J.P. On-orbit servicing: Inspection repair refuel upgrade and assembly of satellites in space. *Aerosp. Corp. Rep.* **2019**, *25*. Available online: https://aerospace.org/sites/default/files/2019-05/Davis-Mayberry-Penn_OOS_04242019.pdf (accessed on 27 July 2025).
9. Mu, Z.; Xu, W.; Liang, B. Avoidance of multiple moving obstacles during active debris removal using a redundant space manipulator. *Int. J. Control Autom. Syst.* **2017**, *15*, 815–826. [CrossRef]
10. Dubanchet, V.; Saussié, D.; Alazard, D.; Bérard, C.; Peuvédic, C.L. Modeling and control of a space robot for active debris removal. *CEAS Space J.* **2015**, *7*, 203–218. [CrossRef]
11. Palma, P.; Rybus, T.; Seweryn, K. Application of Impedance Control of the Free Floating Space Manipulator for Removal of Space Debris. *Pomiary Autom. Robot.* **2023**, *27*, 95–106. [CrossRef]
12. Sampath, S.; Feng, J. Intelligent and robust control of space manipulator for sustainable removal of space debris. *Acta Astronaut.* **2024**, *220*, 108–117. [CrossRef]
13. Cruijssen, H.; Ellenbroek, M.; Henderson, M.; Petersen, H.; Verzijden, P.; Visser, M. The european robotic arm: A high-performance mechanism finally on its way to space. In Proceedings of the 42nd Aerospace Mechanism Symposium, Baltimore, MD, USA, 14–16 May 2014
14. Xu, J.; Liu, X.; He, L. Vision system for robotic capture of unknown non-cooperative space debris. *Nonlinear Dyn.* **2015**, *81*, 845–858. [CrossRef]

15. Liu, Y.; Xu, S.; Wang, K.; Wang, J. Joint perception, planning and control for autonomous robotic capture of non-cooperative space debris. *Acta Astronaut.* **2024**, *215*, 355–369. [CrossRef]
16. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
17. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
18. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
19. Chen, X.; Huang, Q.; Wan, W.; Zhou, M.; Yu, Z.; Zhang, W.; Yasin, A.; Bao, H.; Meng, F. A robust vision module for humanoid robotic ping-pong game. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 35. [CrossRef]
20. Qiao, S.; Shen, D.; Wang, X.; Han, N.; Zhu, W. A self-adaptive parameter selection trajectory prediction approach via hidden Markov models. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 284–296. [CrossRef]
21. Wiest, J.; Höffken, M.; Kreßel, U.; Dietmayer, K. Probabilistic trajectory prediction with Gaussian mixture models. In Proceedings of the 2012 IEEE Intelligent Vehicles Symposium, Madrid, Spain, 3–7 June 2012; pp. 141–146.
22. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971.
23. Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; Alahi, A. Social gan: Socially acceptable trajectories with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2255–2264.
24. Shafiee, N.; Padir, T.; Elhamifar, E. Introvert: Human trajectory prediction via conditional 3d attention. In Proceedings of the IEEE/cvf Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16815–16825.
25. LaValle, S.M.; Kuffner, J.J. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics*; CRC Press: Boca Raton, FL, USA, 2001; pp. 303–307.
26. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
27. Camacho, E.F.; Bordons, C.; Camacho, E.F.; Bordons, C. *Constrained Model Predictive Control*; Springer: Berlin/Heidelberg, Germany, 2007.
28. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
29. Hsu, D.; Sánchez-Ante, G.; Sun, Z. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 3874–3880.
30. Li, Q.; Xu, Y.; Bu, S.; Yang, J. Smart vehicle path planning based on modified PRM algorithm. *Sensors* **2022**, *22*, 6581. [CrossRef]
31. Marini, F.; Walczak, B. Particle swarm optimization (PSO). A tutorial. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 153–165. [CrossRef]
32. Hao, Z.; Zhang, D.; Honarvar Shakibaei Asli, B. Motion Prediction and Object Detection for Image-Based Visual Servoing Systems Using Deep Learning. *Electronics* **2024**, *13*, 3487. [CrossRef]
33. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
34. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
35. Wang, X.; Gao, H.; Jia, Z.; Li, Z. BL-YOLOv8: An improved road defect detection model based on YOLOv8. *Sensors* **2023**, *23*, 8361. [CrossRef]
36. Jocher, G.; Stoken, A.; Borovec, J.; Changyu, L.; Hogan, A.; Diaconu, L.; Poznanski, J.; Yu, L.; Rai, P.; Ferriday, R.; et al. ultralytics/yolov5: V3.0. *Zenodo* **2020**. [CrossRef]
37. Department, M.V.I. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. 2022. Available online: https://github.com/meituan/YOLOv6 (accessed on 1 August 2025).
38. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2023**, arXiv:2207.02696.
39. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
40. Liu, S.; Qi, L.; Qin, H.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
41. Li, Z.; Wu, X.; Yang, S.; Lin, Z.; Hu, X. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 6–12 December 2020; Volume 33, pp. 21002–21012.

42. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. OTA: Optimal Transport Assignment for Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 303–312.
43. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2003.
44. Cross, R. *Physics of Baseball and Softball*; Springer: New York, NY, USA, 2011.
45. Shi, Y.; Eberhart, R.C. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
46. Jiang, Z.; Zhang, Q. Time optimal trajectory planning of five degrees of freedom manipulator based on PSO algorithm. In Proceedings of the 2022 4th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP), Hangzhou, China, 8–10 July 2022; pp. 1059–1062.
47. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. [CrossRef]
48. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
49. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
50. Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T.Y.; Tegmark, M. Kan: Kolmogorov-arnold networks. *arXiv* **2024**, arXiv:2404.19756.
51. Yang, X.; Wang, X. Kolmogorov-arnold transformer. *arXiv* **2024**, arXiv:2409.10594.
52. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. Detrs beat yolos on real-time object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 16965–16974.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

https://dspace.org/

Dynamic space debris removal via deep feature extraction and trajectory prediction in robotic systems

Zhang, Zhuyan; Zhang, Deli; Honarvar Shakibaei Asli, Barmak

2025-08-28
MDPI
https://doi.org/10.3390/robotics14090118
Zhang Z, Zhang D, Honarvar Shakibaei Asli B. (2025) Dynamic space debris removal via deep feature extraction and trajectory prediction in robotic systems. Robotics, Volume 14, Issue 9, August 2025, Article number 118

Downloaded from a DSpace repository. https://dspace.org/