Cross-Spatial Fusion and Dynamic-Range Particle Filter Based FPGA-GPU Architecture for 1-ms RGB-Based Object Pose Tracking

Wei Wang, Yuan Li, Tingting Hu, Ryuji Fuchikami, and Takeshi Ikenaga, *Senior Member, IEEE,*

***Abstract*—Synchronization with real-time data is critical for vision-driven measurement techniques, particularly in factory automation (FA) object pose tracking. Increasing image sampling frequency and reducing processing delays can enhance production efficiency, yet most existing works focus on accuracy rather than speed due to computational complexity. This paper develops a high frame rate and ultra-low delay object pose tracking architecture using FPGA-GPU hetero complementation. It proposes two methods: a cross-space fusion module that enhances feature extraction and depth accuracy by using 3D models and temporal information; A dynamic-range particle filter designed to improve pose calculation accuracy with a hardware-oriented approach. Furthermore, a high-frame-rate dataset for factory automation pose tracking was developed to evaluate the system. The results indicate that compared to the state-of-the-art, the proposed method exhibits only a 0.04% reduction in ADD-0.1d accuracy during frame-by-frame operation but a 22.25% improvement during real-time operation. For an input of a 640×360 image sequence at 1000 FPS, the proposed method processes the entire sequence from camera capture to output with 0.927 ms/frame.**

***Index Terms*—Object pose tracking, FPGA-GPU hetero complementation, real-time processing, particle filter, Ultra-low delay visual system.**

I. Introduction

REAL-TIME refers to a system where computations are performed concurrently with an external process, allowing the results to be used immediately for control, monitoring, or response [1]. In general areas of computer vision [2]–[5], real-time is considered to be a processing rate of 30 or 60 frames per second (FPS). This rate aligns with the video frame rate in life scenarios and matches the frequency of human visual perception. However, for machine-based industrial scenarios, i.e., factory automation, the speed requirement for real-time is higher. Take the most common process of robotic arm gripping objects on production lines as an example. If objects are moving at high speeds, rolling during transfer, or falling, a machine vision system capturing images at 30 or 60 FPS would not be sufficient to measure them accurately. To prevent economic losses from production line stoppages or reduced production speeds, high frame rate (e.g., 1000 FPS) video inputs are necessary. To align these inputs, a system with ultra-low delay processing speed (e.g., 1 ms) is highly expected. Existing methods, due to their general-purpose design and lower processing speeds, are unable to handle high-frame-rate video inputs at the corresponding frequency. This results in significant delays during processing, causing the output to lag behind the real-time state of the input. The output becomes sluggish and falls significantly behind the input. Achieving ultra-high processing speeds with minimal latency is essential to meet the measurement requirements of high-speed industrial scenarios, enabling more timely descriptions of the object's motion.

In industrial production lines, the measurement of an object's pose involves two distinct components: object displacement and object rotation. Object displacement is relatively straightforward to measure due to the intuitive nature of motion. Existing methods [6] and tools (e.g., infrared sensors [7]) are employed for high-speed spatial displacement measurements. Conversely, measuring object rotation demands precise perception and a detailed understanding of the object's structure and surface, which becomes particularly challenging at high speeds. Recent studies have explored rotational pose measurements using instruments like binocular cameras [8] or depth cameras [9]. Particularly, methods [10]–[12] based on RGB-D cameras for pose estimation have been proven to achieve high accuracy. However, these methods require additional processing of depth information per frame, resulting in increased time from measurement to pose computation. This makes it challenging to achieve high frame rates and ultra-low latency in scenarios involving high-speed object rotation. Furthermore, in practical industrial environments—such as the handling of small or complex-shaped parts, pose estimation of transparent plastic or glass objects, high-speed vibrating conveyor belt production, and large-object assembly like automotive production lines—the requirements for measurement precision, operational range, and sampling frequency are significantly higher and necessitate frequent adjustments. In contrast, RGB cameras, due to their broader application scope, meet these industrial demands, offering a more intuitive and cost-effective measurement choice.

Single-view RGB video information presents a potential for ultra-high-speed rotational pose measurements, yet numerous challenges persist. Firstly, most existing image processing works use a Central Processing Unit (CPU) and a Graphics Processing Unit (GPU) as computational cores. The memory-based Neuman architecture determines that each image is read and stored in its entirety. The I/O bottleneck in the memory swapping process increases the measurement latency. On the

Wei Wang, Yuan Li and Takeshi Ikenaga are with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan (e-mail: wangwei_64@toki.waseda.jp; liyuan3104@fuji.waseda.jp; ikenaga@waseda.jp).

Tingting Hu and Ryuji Fuchikami are with Panasonic Connect Co., Ltd., Osaka, 540-8553, Japan (e-mail: hu.tingting@jp.panasonic.com).

other hand, existing work struggles to balance the trade-off between accuracy and speed. Traditional mathematical algorithms, while faster, only utilize partial image feature information for computation, resulting in lower accuracy and higher errors. Recent advances in deep learning algorithms, whether implemented as end-to-end networks or in multi-stage architectures, have achieved high accuracy and shown robustness in various environments. However, these algorithms require substantial computational resources, and both training and inference speeds are generally slow. In addition, numerous studies have utilized techniques such as network quantization [13]–[17], pruning [18]–[20], and knowledge distillation [21] to accelerate algorithms and reduce memory consumption on hardware platforms. However, despite significant speed improvements over existing methods, these approaches still fail to achieve ultra-low latencies, such as a few milliseconds. This limitation is partly due to the large neural network architectures used, which still require substantial processing time, and partly because these studies have not designed and optimized specifically for individual targets or backgrounds. The primary goal of this research is to achieve a high frame rate and ultra-low delay in object pose measurements while ensuring accuracy.

Previous studies [22]–[25] have highlighted the benefits of using Field-Programmable Gate Array (FPGA) for ultra-low delay in image processing. The advantages stem not only from FPGA's capability to bypass software layers and implement customized data paths directly at the hardware level, but also from their capability to support hardware-accelerated structures like pipeline and highly parallel computing. This configuration allows for the synchronous performance of image capture (input), processing, and subsequent computation, thereby significantly reducing the time required for memory storage and swapping. However, the limited on-chip computational resources of FPGA are inadequate for high-level information processing tasks such as object pose estimation. To overcome this limitation, Zhang *et al.* [26] developed an FPGA-GPU hetero complementation strategy for hand detection. This approach involves running a high-accuracy, low-speed deep neural network on the GPU and a low-accuracy, high-speed binary neural network on the FPGA, effectively achieving a balance between speed and accuracy.

To enhance object pose tracking and achieve real-time processing for high frame rate videos, this paper is developed based on Zhang's strategy. On the GPU (PC), a convolutional neural network (CNN) is used to perform a high-accuracy absolute object pose from a single frame, along with other time-consuming processes such as feature points generation and depth computation. Additionally, 3D model information is incorporated to enhance the accuracy of these calculations. On the FPGA, the particle filter, a hardware-oriented temporal algorithm, is employed to conduct high-speed calculations of the relative object pose. The goal of this paper is to achieve millisecond-level (1000 FPS) processing for the whole process from image acquisition to pose output while maintaining accuracy comparable to other mainstream object pose measurement algorithms. The main contributions of this paper are summarized as follows:

1) A hardware-software cooperative architecture for high frame rate and ultra-low delay object pose tracking is proposed. Through a feature points approach based on FPGA-GPU hetero complementation, the two platforms operate synchronously, and the delay caused by running the neural network on GPU is avoided.
2) A cross-space fusion module running on GPU is proposed. This module enhances the 2D feature points selection strategy by integrating timing information and analyzing the Computer-Aided Design (CAD) structural model as prior knowledge, thereby improving the accuracy of feature points depth calculations.
3) A dynamic-range distribution particle filter algorithm is implemented on the FPGA. This algorithm, integrated within the FPGA-GPU structure, dynamically adjusts the distribution range of the particle filter. Experimental results indicate a 25.27% improvement in accuracy compared to existing particle filter algorithms.
4) An object pose instrument that allows objects to rotate at all angles at high speed is fabricated, and the first high frame rate (1000 FPS) video dataset for object pose tracking is constructed. Unlike other existing object position estimation datasets, the camera in this dataset is stationary while the object rotates. The motion patterns of objects in industrial automation scenarios are modeled.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents the proposed method and implementation details. Section IV presents the dataset, experimental results and analyses. Finally, Section V gives the conclusion.

II. Related Works

A. RGB-Based Object Pose Measurement Algorithm

In recent decades, a large number of RGB vision-based object pose measurement algorithms have emerged. Dominant approaches [27] include image-based algorithms that estimate pose for each frame independently and video-based tracking algorithms that utilize temporal information across frames. A comparison of some of the algorithms is presented in Table I.

Image-based pose estimation algorithms calculate the absolute pose of an object by comparing and matching a single picture to a 3D model [28]. Earlier traditional algorithms, such as the Efficient Perspective-n-Point (EPnP) [29], affine transform-based matching [30], and the works utilizing Scale-Invariant Feature Transform (SIFT) descriptors [31]–[33], relied on hand-designed image features for complex mathematical operations. They primarily focused on local keypoints and feature matching. Although these methods were robust in various scenarios, they were heavily dependent on the precision of keypoints and generally less accurate. Lowe *et al.* [34] introduced a method using 3D model alignment, and Li *et al.* [35] applied shape alignment for automotive attitude detection. Some algorithms [36]–[38] also utilized contour and edge information for 2D to 3D matching, targeting simpler 3D shapes and curves. However, their performance degraded with complex models and in diverse scenes.

With the advent of deep learning, numerous neural network-based methods have been developed for object pose estimation. Some methods employ neural networks for direct end-to-end pose estimation. SSD-6D network [39] categorized rotation as a classification problem, while PoseCNN [40] employed quaternion regression for 3D rotation estimation. EfficientPose [41] leveraged the EfficientDet framework to combine high accuracy with fast performance, and networks like GDR-Net [42] utilized geometric features for regression. Despite their simplicity and efficiency, these direct methods often lacked sufficient generalizability for complex scenarios. Other approaches incorporated intermediate information to enhance estimation. BB8 [43] predicted the 2D projection of the bounding box corners of a 3D object post-segmentation, then computed the pose. YOLO-6D [44], leveraging the You Only Look Once (YOLO) detection framework, enabled fast operations due to accurate predictions without the need for post-processing. To improve feature points localization, some methods adopted a pixel-by-pixel voting [45], [46] or mixed information method [47], but these approaches were computationally intensive. DeepIM [48] and the method proposed by Manhardt *et al.* [49] employed fine-tuning-based strategies. They iteratively minimized the gap between real and rendered images, which improved accuracy but was time-consuming and relied on high-quality texture models. In addition to feature points, some approaches [50]–[53] directly utilize dense pixel coordinates as inputs, establishing dense correspondences between 2D images and 3D models through neural networks. By leveraging dense coordinates, these methods effectively handle objects with complex backgrounds or high symmetry. However, due to the need to process a large number of pixel points, such methods typically have high computational costs and longer processing times. In recent years, self-supervised [54] or model-free [55]–[57] approaches have also emerged. Some methods involve an initialization process to model the object beforehand. These methods reduce annotation costs and lessen the reliance on input data. However, they require prior data preparation, which leads to disadvantages in both accuracy and speed. This makes them unsuitable for the industrial automation scenarios discussed in this paper.

Video-based pose tracking algorithms initiate tracking after determining the object's initial pose. Early methods, such as Vacchetti *et al.* [58] proposed, established 2D-3D relationships by comparing feature points across frames, but they were prone to feature points loss and demanded specific surface properties. Methods like RAPID [59] estimated relative poses between frames by analyzing gradients of projected edges. More recent methods combined particle filter with color [60], contour [61], and weight information [62], enhancing both accuracy and speed but struggled with high-speed movements. The first region-based tracking method, PWP3D [63], integrated color segmentation models with object rendering techniques to optimize pose estimation. Subsequent methods employed more refined mathematical approaches, but their practical application was often limited by the need for extensive manual features and parameter definitions.

Recently, deep learning-based RGB tracking methods have gained popularity. For instance, PoseRBPF [64] utilized the particle filter trained on a codebook to estimate pose by comparing real and rendered images, requiring high-quality models for accurate rendering. DeepAC [65], known for its performance, projected the object model onto the image plane for initial contour generation. It then used a lightweight network to adjust the contours to the actual object boundaries, making it suitable for mobile deployment, but not specifically tailored for industrial applications. Lin *et al.* [66] recently proposed a category-level object pose tracking method based on RGB video sequences. This method utilizes a keypoint-based pose tracking approach and introduces uncertainty estimation to improve model robustness. Although this approach no longer relies on precise 3D models as input and achieves relatively high accuracy, the introduction of uncertainty estimation significantly slows down the inference speed, making it unsuitable for the scenarios proposed in this paper.

TABLE I
BRIEF SUMMARY OF SOME STATE-OF-THE-ART OBJECT POSE MEASUREMENT ALGORITHM

Method	Frame Rate	Input	Publication Title	Year
Vacchetti *et al.* [58]	15 FPS	Video	IEEE T-PAMI	2004
PWP3D [63]	20 FPS	Video	IJCV	2012
BB8 [43]	3 FPS	Image	ICCV	2017
YOLO-6D [44]	50 FPS	Image	CVPR	2018
EfficientPose [41]	27.5 FPS	Image	arXiv	2020
PoseRBPF [64]	11.5 FPS	Video	IEEE T-RO	2021
GDR-Net [42]	45.5 FPS	Image	CVPR	2021
DeepAC [65]	25 FPS	Video	ICCV	2023
This work	**1000 FPS**	Video	-	-

B. Hardware-Based Object Tracking Implementation

Although there has been no research focused on object pose tracking from monocular RGB images on FPGA-based hardware platforms [67], they are widely used for acceleration and low-latency processing due to their hardware-connected architecture, particularly in video-based visual tracking algorithms. The optical flow method is a common approach for hardware implementation of object tracking: Barranco *et al.* [68] introduced an optical flow core with a multi-scale scaling method employing a pipeline architecture. Tomasi *et al.* [69] achieved a processing speed of 160 FPS using a Digital Signal Processor (DSP) board as a co-processor for the FPGA atop a pipeline structure. Gultekin *et al.* [70] reached a frame rate of 257 FPS through a pipeline parallel architecture, while Ishii *et al.* [71] achieved a 1000 FPS sampling frequency with their custom-designed H^3 dual FPGA platform. Hu *et al.* [23] implemented LK-optical flow computation at 1000 FPS using a parallel pipeline with high accuracy facilitated by a timing prediction algorithm.

Another hardware-friendly tracking method is the filter. Marzotto *et al.* [72] utilized three independent Kalman filters for lane detection and tracking. Jarrah *et al.* [73] executed particle filter on an FPGA using a pipelined-parallel architecture. Engineer *et al.* [74] introduced a variable-number particle filter for 650 FPS color video tracking. Yang *et al.* [25] achieved 762 FPS in camera pose tracking with an improved particle-filtered

Simultaneous Localization and Mapping (SLAM) architecture. Despite diverse objectives, these efforts demonstrate that FPGA implementation can significantly accelerate tracking algorithms compared to software-based approaches.

Numerous studies have focused on multi-platform cooperation to enhance the integration of hardware and software capabilities. The most widely adopted hetero complementation involves the FPGA-SOC architecture, which utilizes both the programmable logic (PL) and the processing system (PS) components of the FPGA. For instance, Ali *et al.* [75] proposed a meanshift algorithm that leverages hardware-software collaboration. It achieved a frame rate of 290 FPS by running the prediction filter and updating the tracking loop in software, while offloading computationally intensive tasks to the hardware. Similarly, Babu *et al.* [76] implemented a multidimensional Kalman filter for object tracking and motion detection on System on Chip (SOC), enhancing tracking speed to 49 FPS. Additionally, Kosuge *et al.* [77] utilized an FPGA-SOC for deploying an RGB-D based iterative-closest-point (ICP) algorithm for object pose tracking. Hobden *et al.* [78] introduced a neural network-based Unmanned Aerial Vehicle (UAV) tracking method using a collaborative architecture, increasing overall speed to 54.67 FPS by implementing part of the network layer in the PS.

Some approaches extended this integration by directly connecting the FPGA to a PC. For example, Konomura *et al.* [79] achieved 100 FPS in UAV camera attitude tracking through FPGA-CPU co-structuring with visual annotation support. Furthermore, Zhang *et al.* [26] designed an FPGA-GPU hetero complementation for two-handed tracking, where a high-accuracy neural network operated on the GPU and a hardwired binary neural network on the FPGA. This combination of preliminary results from the FPGA with refined outputs from the GPU delivered a final output with high accuracy and an ultra-low delay of 1000 FPS. Although this architecture significantly accelerates the tracking algorithm, its design and target limitations make it suitable only for the position detection of 2D hand targets. For rotating 3D objects, the architecture cannot reference the object's 3D model nor output the object's pose. This limitation prevents accurate estimation of the object's spatial orientation. Therefore, this architecture is not suitable for 3D object pose tracking in factory automation scenarios. Carballo-Hernández *et al.* [80] proposed a method for automatically partitioning convolutional neural network models through geometric programming, enabling load balancing between GPUs and FPGAs. This approach accelerates tasks such as vision-related applications that rely on convolutional neural networks. Zhang *et al.* [81] introduced a hardware-software co-design approach optimized for object detection and tracking tasks in video data. By accelerating low-level image processing tasks on FPGAs and completing high-level inference on CPUs, this method significantly improves system throughput, achieving a processing speed of 314.7 FPS. Wang *et al.* [82] proposed an optimization for parallelization in Advanced Driver Assistance System (ADAS) applications within FPGA-GPU heterogeneous systems, particularly focusing on lane detection and tracking. By leveraging the strengths of both FPGAs and GPUs, this method achieves a 109.21% improvement in processing speed, enhancing real-time object tracking efficiency in ADAS systems. Since existing hardware methods are not specifically targeted at RGB-based object pose tracking, there is a critical need to develop a high frame rate and ultra-low delay object pose tracking architecture tailored to these applications.

C. Particle filter

Particle filter is a widely used architecture in the field of video tracking, particularly valued for its high accuracy and robustness in pose estimation. It involves similarity estimation and resampling steps, which are executed after distributing the results of the previous frame randomly. In recent years, various particle filtering algorithms have been proposed for different tasks. Zhang *et al.* [83] improved tracking robustness through a consistent low-rank sparse representation. Li *et al.* [84] introduced a particle filter utilizing Reliable Patch, which adapts to more complex scenarios, but shows limitations when dealing with pose changes or fast-moving objects. Zhang *et al.* [85] proposed a multi-task correlation filter to enhance the robustness of object tracking. Qiu *et al.* [86] developed an improved cuckoo search particle filter, increasing both the accuracy and efficiency of the particle filter. Lin *et al.* [87] enhanced estimation accuracy by adaptively adjusting particle weights. In the field of object pose tracking, Deng *et al.* [64] utilized a Rao–Blackwellized particle filter, which performs likelihood estimation by comparing full-pixel decoder outputs with rendered images, achieving high-precision object pose tracking. Although these methods demonstrate advantages in handling complex tasks, their computational complexity makes them difficult to implement on hardware acceleration platforms and unsuitable for the speed requirements of this work. Therefore, this work only employs 2D feature points and 3D projection points for likelihood estimation.

III. Proposed Method

A. Overview

The hardware-software collaborative architecture proposed for high frame rate and ultra-low delay object pose tracking is illustrated in Fig. 1. Similar to other model-based object pose tracking systems, this architecture processes video and CAD model as inputs. However, it is distinct in targeting ultra-high speed rotational motions of objects in factory automation scenarios, as outlined in Section I. The system utilizes a high-speed camera capable of capturing video streams at a frequency of 1000 frames per second, ensuring real-time synchronization between camera operation and online processing. The video stream is transmitted to the FPGA, where it undergoes pipeline processing in pixel-stream format. This allows processing to begin before the completion of a frame capture, ensuring seamless integration of the capture and processing stages. Simultaneously, the image data processed by the FPGA is further transmitted to the GPU platform for subsequent image analysis and processing, meeting the demands for efficient and low-delay real-time image processing. The CAD model input is a 3D structural model of the object, represented as a textureless point cloud, which is pre-owned

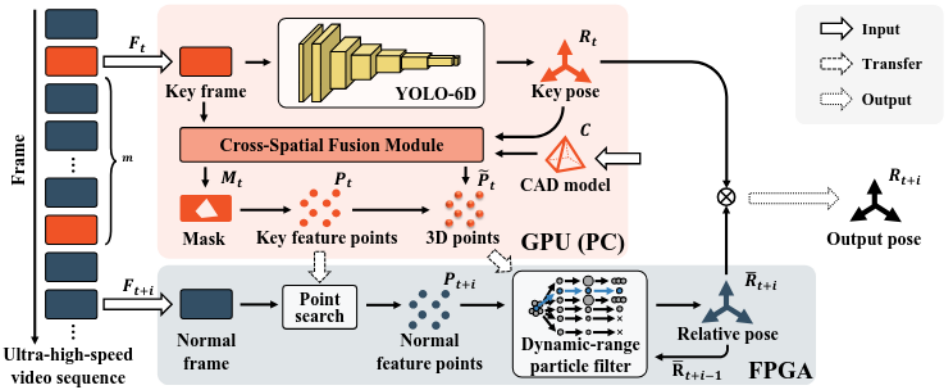

Fig. 1. Framework of the proposed ultra-low delay object pose tracking method. By passing the feature points data in GPU (PC) and FPGA, the delay due to the neural network operation is avoided, and the computation of relative pose is performed in a hardware-oriented way.

during factory production. Following Zhang *et al.* [26], the FPGA-GPU hetero complementation architecture is adopted. The high frame rate video input is divided into key frames and normal frames. Key frames run on the GPU (PC) and perform computationally intensive and time-consuming tasks. These tasks include absolute pose estimation using a network, as well as feature points detection and depth calculations. Normal frames are processed on the FPGA. Relying on its pipeline and highly parallel architecture, the FPGA achieves ultra-low delay in producing frame-by-frame relative pose outputs. In this architecture, one key frame is set for every m normal frames. The value of m is determined by the sum of the neural network's single-frame inference time, the running time of the remaining GPU computations, and the data transfer time from the GPU to the FPGA.

It is assumed that key frame F_t occurs at the moment t. Results are available at $t+m$ following computation by the GPU, marking the next key frame. Normal frames from $t+m$ to $t+2m$ utilize the key frame at t as a baseline, where a normal frame at this interval is denoted as F_{t+i}. For key frames, the neural network performs absolute pose inference first. YOLO-6D [44], is known for its simple structure and high speed in pose measurement. To minimize the key frame interval m and enhance accuracy, YOLO-6D is chosen as the backbone network. Upon processing input F_t, the network outputs a rotation matrix R_t, representing the absolute pose of the object. This matrix, based on the initial pose of the input model, is subsequently used to rotate the 3D model point clouds C, aligning them with the object's position at the current frame. This process generates the transformed 3D model point clouds $\widehat{C}_t$ at moment t, as shown in the following equation.

$$\widehat{C}_t = R_t \cdot C. \tag{1}$$

Then, $\widehat{C}_t$ and F_t are input into the cross-space fusion module. This module is designed to establish a mutual indexing relationship between 2D images and 3D point clouds, denoted as $\mathscr{S}_{2D\Leftrightarrow 3D}$. Its detailed composition and rationale will be explained in detail in Section III-B. The output consists of two parts. Firstly, a partial 3D points index I'_t is projected based on the time series. The feature points generation mask M_t for F_t is obtained by projecting it into 2D space. Afterward, F_t is aligned with the pre-ordered key frame, and N feature points P_t are generated within the mask M_t using the ORB algorithm [88]. Secondly, the corresponding 3D points $\widetilde{P}_t$ on the point clouds are identified by mapping the 2D feature points P_t through the module $\mathscr{S}_{2D\Leftrightarrow 3D}$. The GPU then transfers the coordinates of 2D feature points P_t and 3D points $\widetilde{P}_t$ to the FPGA for processing in normal frames.

For the normal frame F_{t+i}, a pixel stream-based matched feature points search is performed using the method proposed by Hu *et al.* [24]. Based on the key frame F_t and the current frame F_{t+i}, the feature points P_t corresponding to the current frame P_{t+i} are found. Subsequently, the 3D points $\widetilde{P}_t$ and feature points P_{t+i} transmitted by the GPU, are input to the dynamic particle filter module. This module constructs a hardware-oriented particle filter for likelihood estimation using feature points. The details will be discussed in Section III-C. A pose estimation for the current frame is generated based on the results of the previous frame $\overline{R}_{t+i-1}$. The relative pose $\overline{R}_{t+i}$ of the current frame F_{t+i} based on the key frame F_t, is obtained through weighted averaging according to the likelihood weights. The final pose result for the current frame is calculated by multiplying the absolute pose R_t of the key frame with the relative pose of the normal frame, i.e.,

$$R_{t+i} = \overline{R}_{t+i} \cdot R_t. \tag{2}$$

The FPGA hardware design and implementation for processing normal frames will be detailed in Section III-D.

B. Cross-spatial fusion module

In order to reduce the low accuracy caused by the time of neural network computation, a feature points based structure as shown in Section III-A is proposed. Nevertheless, relying solely on 2D feature point detection introduces several challenges. On the one hand, during high-speed rotational pose tracking, selected feature points may become occluded by the object itself. Existing feature points tracking methods measure the matching degree of feature points by calculating the Hamming distance between descriptors [88]. However, it requires corner detection in every frame and a large number of alternative feature points to avoid losing some of the feature points.

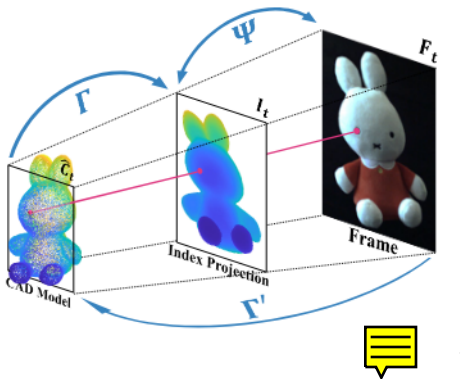

Fig. 2. The proposed cross-space fusion module.

This requirement increases computation and storage demands, which contradicts the need for high-speed processing. On the other hand, pose estimation using feature points requires two frames with significant parallax for triangulation calculations [89]. Under high-speed motion, variations in the selection of parallax reference frames result in decreased accuracy of depth calculations and produce only relative results lacking scale. To address these issues, the cross-space fusion module is proposed. This module leverages 3D model information in the computation, complementing data from 2D images and 3D models to circumvent the aforementioned accuracy degradation. Current model-based pose estimation algorithms typically leverage the geometric information from input CAD models as part of network training or directly match the entire 3D model with 2D images. Such methods maximize the use of 3D model information to enhance accuracy. However, using neural networks, especially end-to-end networks, significantly increases computational costs and reduces processing speed. To minimize the interval between key frames while utilizing 3D information without excessive time consumption, this module introduces a traditional graphics-based algorithm. To establish the connection between 3D and 2D spaces, the proposed approach draws on rasterization algorithms [90], [91] from computer graphics to perform planar projections of 3D models. Additionally, to better highlight the characteristics of point cloud models, recent point-based rendering techniques [92], [93] are adopted. Instead of projecting triangles, point-wise projection is employed. This allows for faster projection while distinguishing the front-back positional relationships. Unlike conventional graphics algorithms, the proposed module also uses point sequences as projection indices. After the 3D points are projected into the 2D space, reverse indexing from the 2D points back to the 3D space is performed, completing spatial fusion and improving model accuracy.

The proposed cross-space complementary module $\mathscr{S}_{2D\Leftrightarrow 3D}$ is illustrated in Fig. 2. This module establishes a mutual indexing relationship between 2D frames and 3D models. To facilitate this, an intermediate sequence projection layer I_t is implemented to assist in determining the 3D point coordinates. The transition from the 3D rotated point clouds $\widehat{C}_t^q$ to the key frame F_t involves two essential steps: projection Γ and mapping Ψ. The process for the projection Γ is described as:

$$I_t(x,y) = \Gamma\left(\widehat{C}_t\right) = \mathrm{argmin}_Z\left(\widehat{C}_t^q\right), \tag{3}$$

$$\widehat{C}_t^q \in \left\{\widehat{C}_t \mid \left\|(x,y) - K\cdot\widehat{C}_t^q\right\|_1 < l\right\}, \tag{4}$$

K is the camera internal reference matrix. The expression $K\cdot\widehat{C}_t^q$ projects 3D coordinates onto the 2D plane, transforming them into 2D coordinate points. The variable l represents the radius of coverage around each projection point, typically set to a few pixels. The subset $\widehat{C}_t^q$ represents a section of the original rotated point clouds $\widehat{C}t$. For each 2D coordinate (x,y) in the sequence projection layer I_t, there are q 3D points that satisfy its 2D projection coordinate. Specifically, $K\cdot\widehat{C}t^q$ is within a Manhattan distance of less than l from (x,y). From the set of points that meet this criterion, the point with the smallest Z coordinate in the original set of 3D points is selected as the value for I_t at the current position. In the process, $\widehat{C}_t$ is sorted by depth, and its index is saved. Subsequently, it is projected onto the 2D plane in a sequence from back to front based on depth. Each projection point, along with its surrounding square area of radius l, is assigned the ordinal value of its depth index. During the projection process, the point cloud projection from the front continuously overlays the back, ensuring depth precedence. After generating the index I_t, it undergoes inter-mapping Ψ with the 2D frame image, which is described as:

$$\Psi(I_t, F_t) : \mathcal{A}(I_t) = \alpha(\mathcal{A}(F_t)), \tag{5}$$

where $\mathcal{A}$ represents the image region area and α denotes the scaling function. Through the use of threshold segmentation and graphical methods, the area size of the object's region in the 2D frame F_t is extracted and scaled to match the size of the region in the index I_t. This process facilitates bidirectional finding and indexing, ensuring that the mapped regions between 2D and 3D accurately correspond in size and orientation.

The method mainly presents two advantages. Firstly, the visual relationships within the 3D point clouds are identified through a back-to-front region projection, which is more efficient than the rasterization method that relies on vector computations. This efficiency stems from the method's requirement to traverse the point clouds only once. Secondly, by generating an ordinal index, it not only facilitates the projection from 3D to 2D but also identifies corresponding 3D points from the 2D frame F_t in reverse. As explained earlier, once the 2D feature points P_t are generated, the corresponding 3D points $\widetilde{P}_t$ are identified using the inverse transformation Γ' of the projection, which proceeds as:

$$\widetilde{P}_t = \Gamma'(P_t) = \widehat{C}_t^{\alpha(P_t)}. \tag{6}$$

Alternatively, the cross-space complementary module is utilized in conjunction with temporal prediction to generate a mask M_t of 2D feature points, which is denoted as:

$$M_t = \Gamma\left(\widehat{C}_t^{\Gamma\left(\widehat{C}_{t+m}\right)}\right)/\alpha. \tag{7}$$

Fig. 3 illustrates the basic process for generating a mask in the

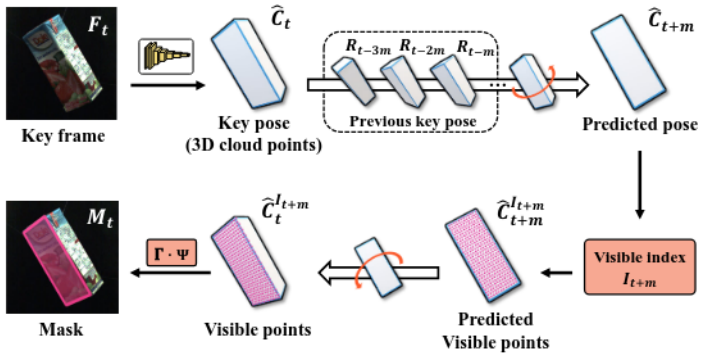

Fig. 3. A framework of 2-D feature point mask generation using the proposed cross-space fusion module combined with temporal prediction. The orange part uses the cross-space fusion module.

GPU portion by combining the cross-space fusion module with temporal prediction. As part of Fig. 1, this process uses the key frame pose R_t output by the YOLO-6D network at time t as input. After combining with the 3D CAD model, the key frame F_t is converted into a 3D point cloud model $\widehat{C}_t$. Based on the results of the preceding two to four key frames, the future key frame pose at time $t+m$ is predicted. Since the purpose of this part is to highlight the object's motion trend rather than to provide precise future rotation predictions, a simple fitting method is used to predict the rotation component. This yields an approximate future pose R_{t+m}, and the 3D model $\widehat{C}_{t+m}$ is obtained by rotating the point cloud. Subsequently, the point cloud is layered and projected using the Γ function from the cross-space fusion module, producing the index of visible 3D points at time $t+m$, denoted as I_{t+m}. Using this index, the visible point set $\widehat{C}_{t+m}^{I_{t+m}}$ of $\widehat{C}_{t+m}$ is identified. This subset is then rotated back to the current time to obtain the visible point set at time t, denoted as $\widehat{C}_t^{I_{t+m}}$. The Γ function is applied again to this point set to prevent partial occlusion of visible points during the rotation process. After generating a new index, the coordinates in the indexed region undergo the Ψ transformation from the cross-space fusion module, which divides by the scaling factor α between the 2D and 3D spaces. This yields the 2D feature points and generates the mask M_t. The actual generated mask image is shown in Fig. 4. Without the mask, feature points may be occluded by the object itself during rotation, leading to matching errors. However, feature points within the mask are still located at subsequent times, effectively improving the accuracy of the feature point matching process.

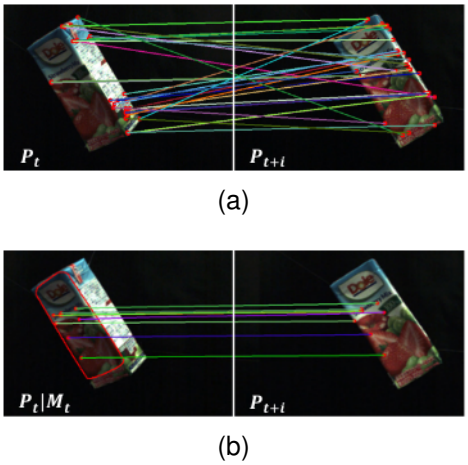

Fig. 4. Feature points extraction results. (a) P_t without prediction mask M_t. (b) P_t with prediction mask M_t.

C. Dynamic-range particle filter

Compared to other filters, such as Kalman filters, particle filters not only handle more complex and nonlinear systems through a non-parametric approach based on importance sampling, but also avoid complex matrix operations and linearization steps, making them easier to implement on hardware platforms. Combined with the key frame-regular frame structure described in Section III-A, a hardware-friendly dynamic-range particle filter is proposed. The procedure of which is detailed in Algorithm 1. As detailed in Section III-A, the inputs for the proposed dynamic distribution range particle filter include the feature points P_{t+i} of F_t, the corresponding 3D points $\widetilde{P}_t$ using $\mathscr{S}_{2D\Leftrightarrow 3D}$, and the results from the previous frame $\overline{R}_{t+i-1}$. First, the current frame particle distribution $\mathbb{P}\left(\overline{R}\right)_{t+i}$ is generated based on $\overline{R}_{t+i-1}$. This distribution follows a dynamic distribution function ϕ, producing a total of J particles. The distribution function ϕ, which follows a uniform distribution centered at $\overline{R}$, is expressed as:

$$\phi(r) = \frac{1}{2\sigma_d}, \overline{R} - \sigma_d \leq r \leq \overline{R} + \sigma_d, \tag{8}$$

where σ_d represents the dynamic distribution range:

$$\sigma_d = \max\left(\beta \cdot \sigma_{\mathbb{P}^*(\overline{R})}, \sigma_{min}\right), \tag{9}$$

and β is the standard deviation scaling factor. Normally, a Gaussian distribution is better suited for simulating and predicting object motion. However, generating high-granularity Gaussian distributions is computationally challenging on hardware. To achieve particle distribution with low hardware resource consumption and high speed, a uniform distribution is adopted, where particles are evenly distributed within the range of σ_d.

To compensate for the information loss caused by the

Algorithm 1 Dynamic-range Particle Filter

input:
P_{t+i}: The feature points of the current frame.
$\widetilde{P}_t$: The 3D points of the key frame F_t.
$\overline{R}_{t+i-1}$: The relative pose of the previous frame.
output:
$\overline{R}_{t+i}$: The relative pose of the current frame.
begin
$\mathbb{P}(\overline{R})_{t+i} \sim \phi(\overline{R}_{t+i-1})$
for $\mathbb{P}(\overline{R})^j_{t+i} \in \mathbb{P}(\overline{R})_{t+i}$ **do**
$\omega^j_{t+i} \leftarrow \mathcal{L}(\widetilde{P}_t, P_{t+i}, \mathbb{P}(\overline{R})^j_{t+i})$
end
$\mathbb{P}^*\left(\overline{R}\right)_{t+i} \leftarrow Resample\left(\mathbb{P}\left(\overline{R}\right)_{t+i}, \omega_{t+i}\right)$
$\overline{R}_{t+i} \leftarrow \frac{1}{J}\sum_{j=1}^{J} \mathbb{P}^*\left(\overline{R}\right)^j_{t+i}$
end

uniform distribution and to align with the proposed key frame-regular frame structure, a dynamic distribution range σ_d is introduced, which references the range of the resampled particle distribution from the previous frame, $\sigma_{\mathbb{P}^*(\overline{R})}$. For regular frames, σ_d is scaled by a reduction factor based on the result of the previous frame, thereby accelerating the contraction of the particle filter distribution range and improving estimation accuracy. To prevent the particles from shrinking completely to a single point, σ_d is not allowed to be smaller than the minimum standard deviation σ_{min}. This approach reduces the number of redundant edge particles generated by the uniform distribution, improving accuracy, but may lead to accumulated errors due to overly rapid shrinkage. Therefore, at key frames, σ_d is re-expanded to its initial size to prevent the true value from falling outside the distribution range when switching reference frames. This adjustment ensures that even if the particle distribution range is exceeded due to the rapid rotational movement of the object, it will be encompassed again after the next key frame.

Next, for each particle in the distribution $\mathbb{P}\left(\overline{R}\right)_{t+i}$, the weight ω is calculated based on the likelihood function $\mathcal{L}$. The likelihood estimation function $\mathcal{L}$ for each particle is given by:

$$\mathcal{L}\left(\widetilde{P}_t, P_{t+i}, \mathbb{P}\left(\overline{R}\right)^j_{t+i}\right) = \frac{1}{\left(\sum_{n=1}^{N}\left\|K \cdot \mathbb{P}\left(\overline{R}\right)^j_{t+i} \cdot \widetilde{P}^n_t - P^n_{t+i}\right\|_1\right)^3}. \quad (10)$$

where K is the camera intrinsic matrix, and $K \cdot \mathbb{P}\left(\overline{R}\right)^j_{t+i} \cdot \widetilde{P}^n_t$ represents the projection of the 3D point $\widetilde{P}^n_t$ rotated by the newly generated particle distribution onto the 2D plane. This process is similar to solving the reprojection error. However, for better hardware implementation, the Manhattan distance between the projected points and the 2D feature points is used instead of the commonly employed Euclidean distance. In selecting the exponent for the likelihood function, different algorithms adjust the distance error exponent based on the scenario. If the exponent is too small, the weight difference between particles becomes too narrow, making it difficult to distinguish the better-performing particles. As new particles are generated in each iteration, insufficient distinction reduces the prediction accuracy of the filter. Conversely, using a higher exponent causes a few particles to dominate the weights, with others approaching zero, which may lead to local optima and reduced algorithm robustness. Additionally, in hardware implementations, each additional calculation during fixed-point arithmetic requires more intermediate storage resources. Therefore, choosing a lower exponent helps with hardware implementation. Considering these factors, the system uses the cube of the distance error for all particles as part of the weight calculation, allowing for sufficient distinction with fewer hardware resources.

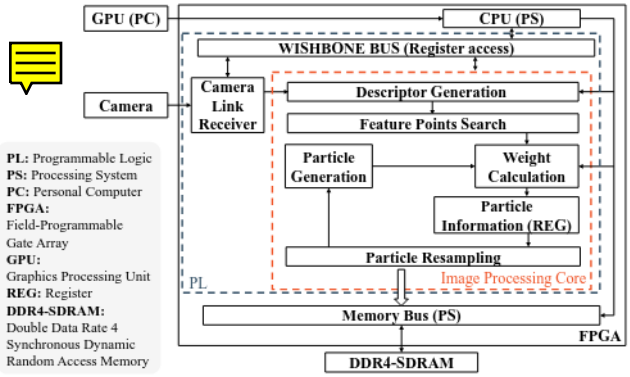

Fig. 5. Overall hardware architecture of the proposed object pose tracking.

Independent random sampling offers high computational efficiency and low hardware resource consumption under reasonable precision and data distribution. Therefore, an independent random sampling approach is applied to the particles. By generating intervals of random numbers, particles are reselected, resulting in the resampled particle distribution $\mathbb{P}^*(\overline{R})_{t+i}$. Finally, since the particle weights are uniform after resampling, the particles are averaged to obtain the relative pose estimate $\overline{R}_{t+i}$ for the current frame. Typically, for particle filters, the resampled particle states are stored and iterated using the state prediction equation to obtain the particle states for the next time step. However, this approach requires significant storage space and access time, making it unsuitable for high-speed stream processing on FPGAs. Therefore, the proposed particle filter uses only the weighted average result of the resampled particles from the previous frame as input and dynamically redistributes particles in the new frame.

D. Hardware Implementation

The hardware architecture of the overall system described in Section III-A is depicted in Fig. 5. To ensure real-time processing synchronization, the hardware system comprises a system-on-chip FPGA, a high-speed camera, and connections to a personal computer (PC) and a GPU. Following the execution of GPU neural network inference and software algorithms on the PC, the absolute pose of the object, 2D coordinates of feature points, and 3D points data are transmitted to the FPGA via a USB3.0 interface. The FPGA features both a processing system (PS) and programmable logic (PL). The

TABLE II
BRIEF SUMMARY OF SOME OBJECT POSE MEASUREMENT DATASETS

Dataset	Data Type	Dynamic Object	For Tracking	Full Area Visible	Frame Rate	Objects	Annotation Method
LINEMOD [94]	real	×	×	×	–	13	Marker
T-Less [95]	real	×	×	×	–	30	Marker
YCB-V [40]	real	×	✓	×	30 FPS	21	Marker
OPT [96]	real	×	✓	×	–	6	Marker
RBOT [97]	semi-synthetic	✓	✓	✓	–	18	Pre-generation
BCOT [8]	real	✓	✓	×	60 FPS	20	Binocular computing
This Dataset	real	✓	✓	✓	1000 FPS	8	Pose sensor

PS functions as an initializer and post-processor, facilitating data transfer from the PC to the PL via the WISHBONE-BUS. The final output from the PL is stored in DDR4-SDRAM for simulation and subsequent analysis. The PL serves as the primary processing unit of the tracking system, receiving a pixel stream from the high-speed camera. This grayscale pixel stream, comprising ten pixels at a time, is transformed into a progressive pixel stream at a frequency of 300 MHz by the camera link receiver, which then serves as input to the image processing core.

The image processing core is divided into two main functions. The first involves searching for normal frame feature points. With the key frame feature points locations received from the WISHBONE-BUS, the surrounding area of a descriptor is generated, and the point with the smallest Hamming distance is identified as the current frame feature point. Since feature point selection occurs on the PC, there is no need for feature points matching between frames on the FPGA. This search process occurs concurrently with the reading of the high-speed pixel stream. The second function, as outlined in Section III-C, involves dynamic-range particle filter after all feature points of the current frame are identified. The process begins by generating random numbers that conform to the dynamic-range distribution and creating corresponding object pose estimation particles. The weights of each particle are determined by projecting the 3D point clouds through rotation and comparing them with the 2D feature points. To reduce the hardware resources consumed by the particle filter, Euler angles are used to store the 3D rotation information. The rotation sequence is Z-Y-X (i.e., Yaw-Pitch-Roll). This transformation is unidirectional and is only used in subsequent steps to convert the angles into a rotation matrix for multiplication with 3D points. Each particle's data, including 3D Euler angle rotation information, individual particle weights, and cumulative particle weights, are temporarily stored in registers in 16-bit signed fixed-point format. Based on the cumulative weights, random numbers are regenerated, and particles are resampled using the roulette method. After weighted averaging, the results are fed back into the particle filter for processing in the next frame and are sent to the memory bus in the PS for pose calculation and output.

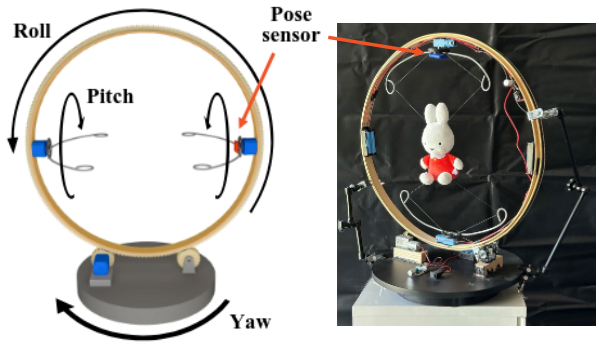

Fig. 6. Object pose instrument. (a) Structural model. (b) Physical device.

IV. EXPERIMENTAL RESULTS

A. Datasets

To evaluate the performance of the proposed method, experiments are conducted using various pose measurement datasets. Most existing datasets, as illustrated in Table II, capture the relationship between the object and the scene. For instance, datasets such as LINEMOD [94], T-Less [95], YCB-V [40], and OPT [96] employ a methodology where the object remains stationary while the camera moves around it. Although this type of movement mimics object rotation, the algorithm inevitably references the surrounding scene, including any markers used for labeling, which may not accurately represent the object's independent motion patterns. The BCOT dataset [8] is a multi-object dataset featuring both indoor and outdoor scenes. It employs unconventional methods such as hands, glass rods, or lines to induce scene-independent motion in objects, with dual cameras utilized for labeling. However, this dataset does not allow objects to be viewed from all angles as it includes at least one support direction. The RBOT dataset [97] employs a semi-synthetic approach, where a 3D model of the object is rendered against a real background, enabling full object perspectives to be displayed. However, there remains a discrepancy between the synthesized and real data. Furthermore, the datasets mentioned above are limited to a maximum frame rate of 60 FPS. As discussed in Section I, while this frame rate is adequate for general pose measurement tasks, it falls short of the requirements for industrial applications, where higher frame rates are necessary to capture more dynamic and rapid movements.

A high frame rate object pose tracking dataset tailored for the factory automation domain is proposed, with the design of the pose tracking instrument depicted in Fig. 6. This figure includes a structural model drawing on the left and

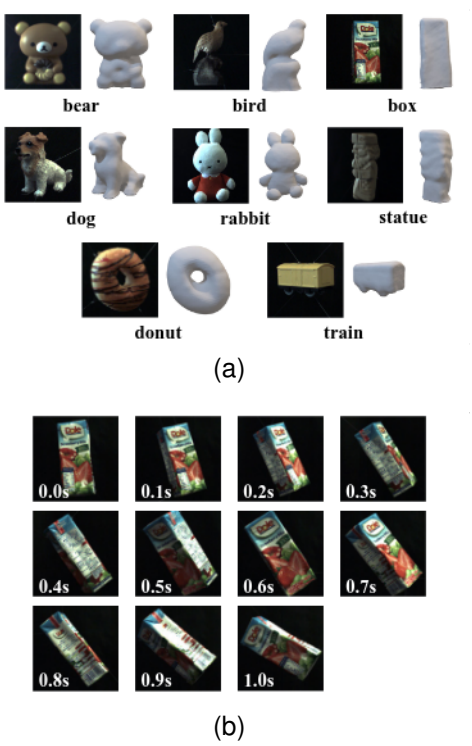

Fig. 7. The proposed high frame rate object pose measurement dataset. (a) Object with its 3D model. (b) Schematic of the motion speed.

a physical drawing on the right. The device employs four motors (one of which is a rotatable base) to facilitate the three-axis rotational motion of the object. The two motors on the innermost symmetric axes are of model TT-130. Each motor is powered by two 18650 batteries, providing a 7.4V voltage. The maximum no-load speed exceeds 310 Revolutions Per Minute (RPM). Speed adjustment is possible through a controller knob. The drive motor for the outer wooden wheels is a JGY-370 DC motor, powered by four 18650 batteries, with a no-load speed exceeding 150 RPM, and speed can also be adjusted. The outermost turntable is an electrically rotatable display platform, with the rotation speed adjustable between 22 seconds and 60 seconds per full rotation. During dataset capture, there are two light sources: natural light coming from the right front of the object through a window, and the light illuminates the object from above. A high-strength transparent wire connects the object at four points in the center of the device, ensuring a stable display from all angles. The background is solid black, designed to emulate the typical environment of an industrial assembly line. The object is configured to rotate at high speeds in random, out-of-plane motions. A pose sensor mounted on the innermost ring of the device captures the rotational pose of the object. Quadratic linear interpolation is employed to supplement the sampling data, ensuring that the sampling frequency aligns with the frame rate of the high-frame-rate camera. To minimize errors from the sensor, interpolation inaccuracies, and drift, a virtual environment the same as the actual filming conditions is constructed using Blender, along with a virtual camera configured to match the parameters of the real high-frame-rate camera. The object's rotational pose is manually adjusted frame by frame to ensure that the rotation of the object in the virtual environment matches the actual frames captured, thereby minimizing errors introduced by the sensor and other experimental settings.

The proposed dataset is shown in Fig. 7a and contains a total of eight objects. 3D structural models of their objects are acquired using a 3D scanner. Each model contains three video sequences, each containing 1000 frames with a frame rate of 1000 FPS and a resolution of 640×360. The motion speed of the objects is shown in Fig. 7b, where each frame corresponds to a real-world time of 1 ms between two adjacent frames. To emphasize the measurement of the object's rotation, the proposed dataset aligns the object displacement for each frame. The rest of the dataset settings are the same as the LINEMOD dataset.

B. Implementation Details

The proposed object pose tracking system consists of an object pose instrument described in Section IV-A and an industrial high frame rate camera, FPGA, and a PC with CPU and GPU. The camera model used is the BASLER acA2000-340, which is an Area Scan Camera with a 2/3" ams CMV2000 CMOS sensor. After configuration, it can output pixel streams at 640×360 resolution with 1000 FPS. The lens used is the Ricoh Lens FL-CC0614A-2M, with an aperture of F1.4 and a focal length of 6 mm. The FPGA model is the Xilinx ZCU104, which includes a Zynq UltraScale and an MPSoC model XCZU7EV-2FFVC1156. In the processing system (PS), there is a quad-core Cortex-A53 64-bit processor and a dual-core Cortex-R5 real-time processor. It contains 504K system logic cells and 461K CLB flip-flops. The rated operating voltage is 12 V, with a working frequency of 300 MHz. The CPU is an AMD Ryzen 9 7950X 16-core processor running at 3.0 GHz with 64 GB of memory. The operating system is Ubuntu 20.04.5 LTS. The GPU is an NVIDIA GeForce GTX 1080 Ti with 11 GB of GDDR5X memory and a thermal design power (TDP) of 250 W, driven by CUDA 11.8. During dataset acquisition, the IMU sensor LPMS-B2 was also used for pose measurement. The resolution is less than 0.05°, with a dynamic accuracy error of less than 2°. The maximum sampling frequency is 400 Hz. The virtual environment construction and error adjustment are performed using Blender 4.3.2. Logic synthesis and hardware implementation were carried out using Vivado 2017.2.

The parameters for the experiment are set as follows: the number of feature points is 15; the number of particles in the particle filter is 150; the initial particle distribution range is 30°; and the camera intrinsic matrix is:

$$K = \begin{bmatrix} 436.36 & 0 & 320 \\ 0 & 327.27 & 180 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

C. Evaluation Metrics

Two evaluation metrics are used to assess the performance of the proposed method and other related approaches: the average 3D distance (ADD) score and axis-angle error. The

ADD score is the most commonly used evaluation metric in object pose estimation and tracking [39], [44], [52], [64], [98], [99]. Specifically for model-based methods, it evaluates the result by recovering the estimated and ground truth poses to the 3D point cloud of the object via a pose matrix, then comparing the differences in the point clouds. The ADD score first calculates the average distance e_v between the estimated and ground truth 3D points, defined as:

$$e_v = \frac{1}{c} \sum_{x \in C} \left\| Rx - R^{gt}x \right\|_2, \tag{12}$$

where C represents the set of 3D points in the model, as defined earlier. The variable c denotes the number of points. R and R^{gt} refer to the estimated and ground truth poses of the object, respectively. Since this work primarily focuses on the 3D rotational pose of the object, the translations between the estimated and ground truth values are normalized, and thus e_v does not account for translation errors. As different objects vary in size, the ADD score does not output an absolute distance but instead compares the average point cloud distance to the object's diameter. The diameter is typically the distance between the two farthest points in the object's 3D point cloud. By comparing the result to the object's diameter, a general metric applicable to objects of any size is formed. In line with other works, the ADD thresholds are set to 10% and 5% of the object's diameter, with e_v smaller than these thresholds considered a success, resulting in ADD-0.1d and ADD-0.05d, respectively. Considering the dataset introduced in this paper, the threshold for ADD-0.1d ranges from 1 cm to 2 cm, providing a more relaxed, general evaluation, while the threshold for ADD-0.05d ranges from 0.5 cm to 1 cm, offering a finer level of distinction. Since the ADD score is reported as a success rate, a result closer to 100 (%) indicates higher accuracy.

Another commonly used metric is the cm-degree pose success rate [42], [47], [57], [100]. Similar to the ADD score, this metric measures both the distance and rotational angle between the algorithm's result and ground truth, with the success rate being the final output. However, for this work, where the focus is on rotational pose, the evaluation is performed directly on the angular measurement. The axis-angle error e_r is used to measure the absolute angular error between the estimated and ground truth values, defined as:

$$e_r = \arccos\left(\frac{\operatorname{trace}\left(R^T R^{gt}\right) - 1}{2}\right). \tag{13}$$

where $\operatorname{trace}(R^T R^{gt})$ denotes the trace of the matrix $R^T R^{gt}$. The matrix trace allows the cosine similarity between the two rotation matrices to be computed, and the arccosine function is then used to derive the rotation angle between them. Compared to using Euler angles or quaternions for multi-dimensional comparison, axis-angle error assumes a 3D axis between the true and estimated rotations, thus using a single parameter to measure the angular difference, making it more suitable for assessing rotational errors in 3D pose estimation. Since the rotation angle is a normalized parameter with a maximum value of 180°, this paper directly uses its absolute value as the evaluation metric, rather than a normalized success rate, to provide a continuous and more intuitive measure of the quality of the rotational result. For axis-angle error, values closer to 0 indicate lower error and higher accuracy.

D. Algorithm Evaluation Results

The proposed method is compared with other methods using the proposed high frame rate object pose measurement dataset. For the overall software testing, the dataset mentioned in Section IV-A was used, with the rotational speed of all objects being approximately $450°/s$. The CPU and GPU mentioned in Section IV-B were used as the software testing platform, where different methods were simulated and tested. First, a pure accuracy comparison is performed. For each work, a frame-by-frame run is performed and compared with Ground Truth, the results are shown in Table III. The results show that the performance of the proposed method is basically up to the best of the existing algorithms. Compared to YOLO-6D, the baseline algorithm of this algorithm, the proposed method only reduces the ADD-0.1d and ADD-0.05d by 0.04% and 24.62%, while it improves the axial angle error by 13.5%. Compared to the other two algorithms, the proposed method improved the mean values of ADD-0.1d, ADD-0.05d, and axis-angle error by 38.77%, 37.36%, and 93.30%, and by 52.98%, 54.73%, and 94.24%, respectively. For the standard deviation (Std) of the axis-angle error, since the proposed method uses video as input, there is a cumulative error between each key frame interval. As a result, the standard deviation of this method is higher compared to YOLO-6D. However, due to the lower mean error, the proposed method still exhibits better average performance compared to the other works. Although the aforementioned algorithms perform well on public datasets, their performance is suboptimal on the proposed industrial scene dataset. Most existing methods are designed for high-precision 3D models and 2D image inputs. In pipeline scenarios, where only structural or low-precision texture models are available, the reduced model accuracy leads to a decrease in overall performance. Additionally, in environments with a lack of background information and low illumination, neural network-based methods struggle to estimate object poses due to the absence of environmental cues, further contributing to performance degradation. Most importantly, in industrial automation scenarios involving high-speed object motion, the slower processing speed of existing methods results in significant errors, as the object may have undergone additional rotation before the system computes the pose, leading to a large discrepancy.

In particular, for applications in industrial scenarios, as mentioned earlier, real-time synchronization with real-world motion is extremely important. Therefore, experiments in terms of real-time processing of the algorithms are carried out. Different algorithms output results according to their running speed and frame rate. Algorithms with lower processing speeds neglect numerous input frames during measurement due to lengthy processing times per frame. Furthermore, two comparisons have been included using Ground Truth running at 60 FPS and 120 FPS, respectively, to offer a more intuitive

TABLE III
ALGORITHM EVALUATION RESULTS RUNNING FRAME-BY-FRAME ON ADD AND AXIS ANGLE ERROR

Method	Metric		Bear	Bird	Box	Dog	Donut	Rabbit	Statue	Train	Average
YOLO-6D [44]	ADD(%) ↑	0.1d	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
		0.05d	**100.00**	**100.00**	**100.00**	**92.05**	**100.00**	**100.00**	**100.00**	**100.00**	**99.01**
	Angle error(°) ↓	mean	**0.79**	4.19	3.28	7.09	**4.50**	4.21	4.23	5.89	4.27
		Std	**0.40**	**2.05**	**2.28**	**5.16**	**2.55**	**2.11**	**5.14**	**3.53**	**2.90**
EfficientPose [41]	ADD(%) ↑	0.1d	**100.00**	88.78	**100.00**	**100.00**	0.76	**100.00**	0.00	0.00	61.19
		0.05d	63.26	19.26	71.30	54.46	0.00	87.96	0.00	0.00	37.03
	Angle error(°) ↓	mean	9.87	19.04	11.15	13.56	133.89	8.54	124.80	119.90	55.09
		Std	1.98	27.18	7.81	18.47	37.28	6.38	40.83	35.97	21.99
DeepAC [65]	ADD(%) ↑	0.1d	**100.00**	0.00	0.00	**100.00**	**100.00**	**100.00**	73.82	0.00	46.98
		0.05d	51.22	0.00	0.00	45.62	0.16	60.28	0.00	0.00	19.66
	Angle error(°) ↓	mean	11.70	138.02	109.96	13.44	96.54	10.50	21.53	111.11	64.10
		Std	3.60	29.44	43.26	6.93	59.93	7.17	15.63	33.24	24.90
This work	ADD(%) ↑	0.1d	**100.00**	**100.00**	**100.00**	99.67	**100.00**	**100.00**	**100.00**	**100.00**	99.96
		0.05d	**100.00**	95.29	**100.00**	47.27	66.32	**100.00**	45.28	40.95	74.39
	Angle error(°) ↓	mean	1.33	**3.98**	**2.36**	**5.40**	5.04	**4.03**	**3.32**	**4.09**	**3.69**
		Std	1.70	5.60	4.23	6.82	5.99	5.23	5.87	6.02	5.18

TABLE IV
ALGORITHM AND BASELINE EVALUATION RESULTS RUNNING IN REAL TIME ON ADD AND AXIS ANGLE ERROR

Method	Metric		Bear	Bird	Box	Dog	Donut	Rabbit	Statue	Train	Average
YOLO-6D [44]	ADD(%) ↑	0.1d	**100.00**	87.03	**100.00**	57.23	70.69	**100.00**	53.26	53.48	77.71
		0.05d	**100.00**	43.91	27.73	17.16	19.69	62.77	12.54	9.56	36.67
	Angle Error(°) ↓	mean	1.50	15.79	17.68	22.15	16.91	15.68	19.42	18.05	15.90
		Std	0.95	4.23	5.01	8.01	5.99	3.62	8.58	7.87	5.53
EfficientPose [41]	ADD(%) ↑	0.1d	**100.00**	33.82	20.61	20.78	1.34	84.32	0.00	0.00	32.61
		0.05d	51.24	0.00	0.00	3.11	0.03	8.48	0.00	0.00	7.86
	Angle Error(°) ↓	mean	10.84	35.74	37.40	40.17	130.60	26.91	122.61	124.66	66.12
		Std	2.08	33.39	10.60	12.29	31.66	6.08	35.35	29.49	20.12
DeepAC [65]	ADD(%) ↑	0.1d	97.59	0.00	0.00	16.00	1.84	69.46	9.95	0.00	24.35
		0.05d	25.84	0.00	0.00	2.18	0.04	0.00	0.00	0.00	3.51
	Angle Error(°) ↓	mean	14.13	135.68	94.85	47.46	98.97	29.72	43.21	108.19	71.53
		Std	2.89	25.99	47.88	15.04	51.96	8.17	14.41	32.67	24.88
60 FPS baseline	ADD(%) ↑	0.1d	**100.00**	98.64	**100.00**	74.65	94.86	**100.00**	66.94	88.78	90.48
		0.05d	**100.00**	58.23	46.71	25.54	38.33	96.01	20.83	25.25	51.36
	Angle Error(°) ↓	mean	1.08	11.89	15.53	17.36	12.89	10.91	15.80	12.70	12.27
		Std	0.28	3.19	3.35	4.18	3.22	1.98	5.08	3.23	3.06
120 FPS baseline	ADD(%) ↑	0.1d	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
		0.05d	**100.00**	**98.77**	**100.00**	**75.10**	**96.12**	**100.00**	**67.14**	**89.48**	**90.83**
	Angle Error(°) ↓	mean	**0.53**	5.90	7.67	8.61	6.36	5.40	7.85	6.29	6.08
		Std	**0.14**	**1.63**	**1.71**	**2.15**	**1.63**	**1.03**	**2.55**	**1.66**	**1.56**
This work	ADD(%) ↑	0.1d	**100.00**	**100.00**	**100.00**	99.67	**100.00**	**100.00**	**100.00**	**100.00**	99.96
		0.05d	**100.00**	95.29	**100.00**	47.27	66.32	**100.00**	45.28	40.95	74.39
	Angle Error(°) ↓	mean	1.33	**3.98**	**2.36**	**5.40**	**5.04**	**4.03**	**3.32**	**4.09**	**3.69**
		Std	1.70	5.60	4.23	6.82	5.99	5.23	5.87	6.02	5.18

comparison of the results. The evaluation results are shown in Table IV. The proposed method, capable of real-time performing calculations at 1000 FPS, processes and outputs results for every frame. The results show that its accuracy is higher than all other existing works. It also improves 9.48%, 23.03%, and 69.91% in ADD-0.1d, ADD-0.05d, and axial angle errors, respectively, compared to the 60FPS baseline. When compared to the 120 FPS baseline, it demonstrates slight reductions of 0.04% and 16.44% in ADD-0.1d and ADD-0.05d while enhancing the axial angle error by 39.22%. Notably, most existing object pose measurements do not surpass 60 FPS, highlighting the efficacy of the proposed method in high frame rate and high-speed motion contexts. Additionally, statistical significance tests between other methods and this work are calculated. In the t-tests, it is found that the p-values are less than 0.01 for both frame-by-frame results (Table III) and real-time results (Table IV), including the baseline of Ground Truth. This indicates that variations in keyframe intervals and differences between image and video algorithms result in significant disparities in the output sequences of the various methods. A partial visualization of the results and baselines is presented in Fig. 8. As shown in the figure, the proposed method closely aligns with the Ground Truth in visualization. When compared with methods that can be executed in real-world scenarios (orange-bordered), the proposed method exhibits the smallest error relative to the Ground Truth. Even when compared to methods that are not feasible for real-world implementation (gray-bordered), the results of the proposed

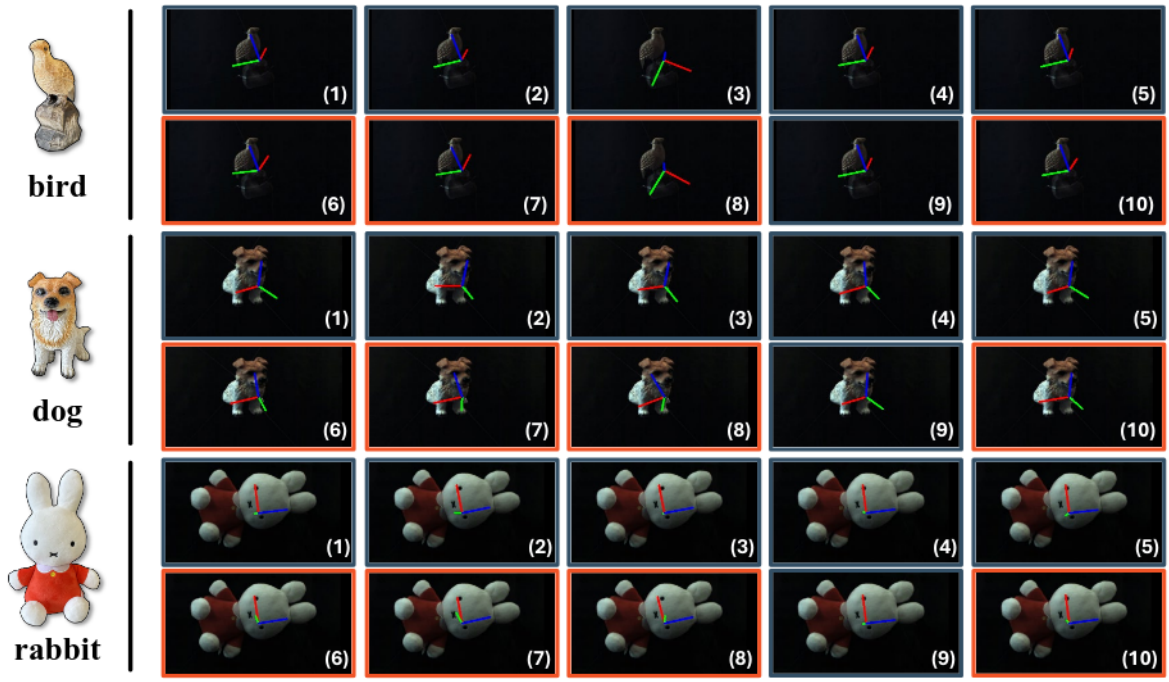

Fig. 8. Examples of visualization result comparisons (bird, dog, and rabbit). The methods used are: (1) YOLO-6D frame-by-frame results. (2) EfficientPose frame-by-frame results. (3) DeepAC frame-by-frame results. (4) 60 FPS baseline. (5) Ground truth. (6) YOLO-6D real-time results. (7) EfficientPose real-time results. (8) DeepAC real-time results. (9) 120 FPS baseline. (10) Results from the proposed method. Methods within gray boxes represent those that cannot run in real-world conditions, while those within orange boxes are capable of real-world execution.

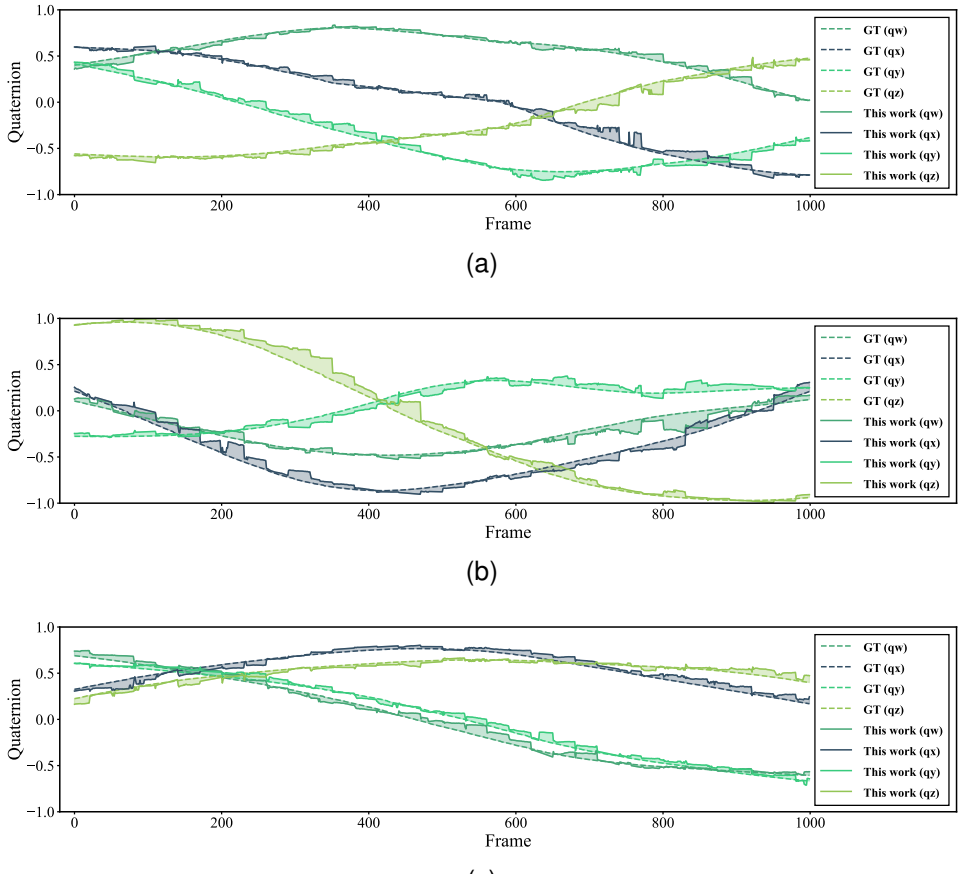

Fig. 9. Comparison between the ground truth object trajectory (dashed line) and the output trajectory from the proposed method (solid line) represented by quaternions. (a) Bird sequence. (b) Dog sequence. (c) Rabbit sequence. The components qw, qx, qy, and qz represent the four elements of a quaternion.

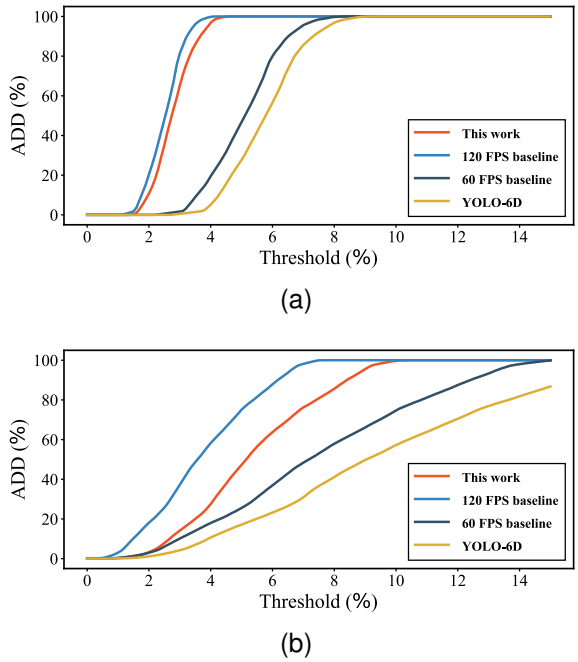

Fig. 10. Comparison of ADD accuracy for different methods under various threshold ratios. The plot includes results from this work, 60/120 FPS baselines, and YOLO-6D running under real-world conditions. (a) Box sequence. (b) Dog sequence.

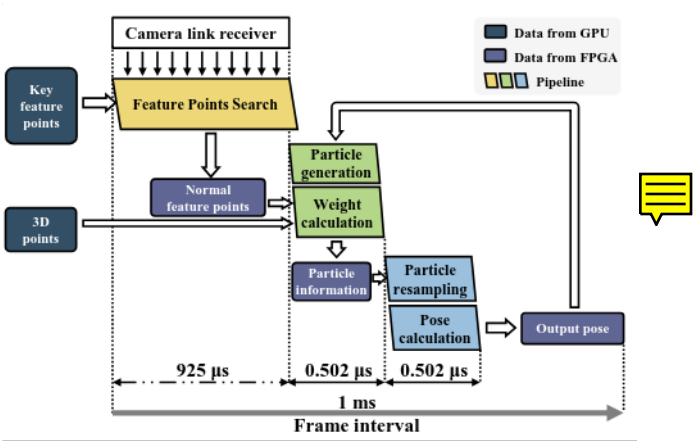

Fig. 11. The timing and data flow within the hardware component of the proposed system. Yellow, green, and blue represent three serial components of the hardware section: feature point search, particle generation, and pose computation. The parallelograms denote pipelined components, where the vertical axis indicates parallelism and the horizontal axis represents execution time. Rounded rectangles represent data either transferred from the GPU or generated and stored within the FPGA.

method outperform several existing approaches and are similar to the 60 FPS and 120 FPS Ground Truth baselines, with only minor deviations. This demonstrates that at the given rotational speed, the proposed method exhibits good performance both in terms of numerical results and visual outcomes.

Fig. 9 compares the true motion trajectory of the object with the output trajectory of the proposed method. Corresponding to the three objects visualized in Fig. 8, each object's motion is displayed over a total of one thousand frames (i.e., 1 second). The rotation of the objects is described using quaternions to ensure the continuity of the motion trajectory. The shaded region between the solid and dashed lines represents the tracking error. As shown in the figure, the trajectory output by the proposed method closely matches the actual trajectory. In cases where larger errors occur, they are corrected at the next key frame, resulting in a staircase-like trajectory in some frame segments.

To mitigate the limitations of using only two ADD diameter ratio thresholds and to demonstrate the effectiveness of the proposed method under all threshold conditions, the curves in Fig. 10 are plotted. These curves compare the ADD success rate with thresholds ranging from 0% to 15% of the object diameter. The figure selects two examples: a simple-shaped object (box) and a more complex object (dog). As shown, the proposed method outperforms the 60 FPS baseline and YOLO-6D, indicating superior tracking performance across all thresholds. Furthermore, the figure highlights that the 5% threshold is a more stringent criterion, effectively showcasing the strengths and weaknesses of different methods. The 10% threshold serves as a standard for distinguishing the difficulty of pose tracking. For objects that are easier to track, most methods achieve a 100% ADD-0.1d success rate, but only a few methods attain a high ADD-0.1d success rate for objects that are more challenging to track.

TABLE V
HARDWARE PERFORMANCE AND RESOURCE UTILIZATION OF THE PROPOSED SYSTEM

Item		Proposed method
Resource utilization	# LUT	198589 (86.19%)
	# LUTRAM	8004 (7.87%)
	# FF	268280 (58.22%)
	# DSP	203 (11.75%)
Performance	Frequency	300 MHz
	Processing time	0.927 ms/Frame

E. Hardware Evaluation Results

The hardware resource utilization for the proposed method, which employs 15 feature points and 150 particles, is detailed in Table V. The system operates at 300 MHz on a Xilinx XCZU7EV-2FFVC1156 model FPGA board, utilizing 198589 Look-Up Table (LUT) resources (86.19% of total) and 268380 Flip-Flop (FF) resources (58.22% of total). The remaining hardware resources are engaged at less than 50%. The system employs a parallel computing and pipelined architecture, processing each frame in 0.927 ms. The timing and data flow of all submodules are illustrated in Fig. 11. The entire hardware system is divided into three submodule pipelines, each performing pipelined processing. The first pipeline includes camera acquisition and feature point search. The high-frame-rate camera captures frames and transfers pixel streams in a total time of 925 μs, with the images transmitted to the FPGA's image processing core via the Camera Link receiver. Similar to the method proposed by Hu *et al.* [24], feature point search is performed concurrently with pixel stream reading. This pipeline outputs the 2D coordinates of the feature points in the current regular frame and temporarily stores them. The search time depends on the location of the

TABLE VI
ABLATION EXPERIMENTS ON THE EFFECTIVENESS OF THE TWO PROPOSED MODULES. W/O CSFM INDICATES THAT THE CROSS-SPACE FUSION MODULE IS NOT USED; W/O DRPF INDICATES THAT NORMAL PARTICLE FILTER IS USED

Configuration	ADD-0.1d(%)↑	ADD-0.05d(%)↑	Angle Error(°)↓
w/o CSFM, DRPF	96.47	45.01	**3.66**
w/o DRPF	99.92	67.88	4.24
w/o CSFM	99.88	70.28	4.47
w/ CSFM, DRPF	**99.96**	**74.39**	3.69

feature points, but typically, feature point search is completed simultaneously with pixel stream reception within 925 μs. The second pipeline, responsible for particle generation, starts immediately after the image is read, within the same 925 μs window, generating pose particles through random number generation and dynamic range adjustment. Since this module only uses pose information from the previous frame, it can tolerate some delays in the feature point search module. Next, 2D feature points and 3D points from the GPU (PC) are used for weight calculation. Once all the weights have been calculated, particle information, including angles and weights, is temporarily stored in the FPGA. Due to the small hardware resource consumption compared to the total FPGA resources, the data can be stored in the FPGA's registers for faster storage and writing. If an excessive number of particles are generated, they are stored in the FPGA's Block Random Access Memory (BRAM). The second pipeline takes a total of 0.502 μs. The particle information is then input into the third pipeline for re-sampling and pose computation. This step requires generating random numbers again and randomly selecting new particles, necessitating a read-back of stored particle information. After resampling, the particles are weighted and averaged to obtain the relative pose. This pipeline also takes 0.502 μs. The result from the third pipeline serves as the final object pose output, and the result is stored for particle generation in the next frame. Experimental results demonstrate that the proposed method achieves real-time processing at 1000 FPS, with efficient hardware resource utilization, completing the entire process from camera reception to pose measurement in under 1 ms per frame.

F. Ablation Study

Ablation experiments are performed to demonstrate the effectiveness of the method proposed in Section III. The effectiveness of the cross-space fusion module (CSFM) of Section III-B and the dynamic-range particle filter (DRPF) of Section III-C are verified with the number of feature points of 15 and the number of particles of 150, respectively. The results are shown in Table VI. The experiments demonstrate that, compared with the normal particle filter, the use of the dynamic-range particle filter improves the ADD-0.1d and ADD-0.05d by 3.42% and 25.27%. The use of the cross-space fusion module improves 0.08% and 4.11%, while reducing the axial angle error by 17.46%. Compared to the structure output using only the basic framework, the increased axis-angle error when using two modules may be due to incorrect matching or

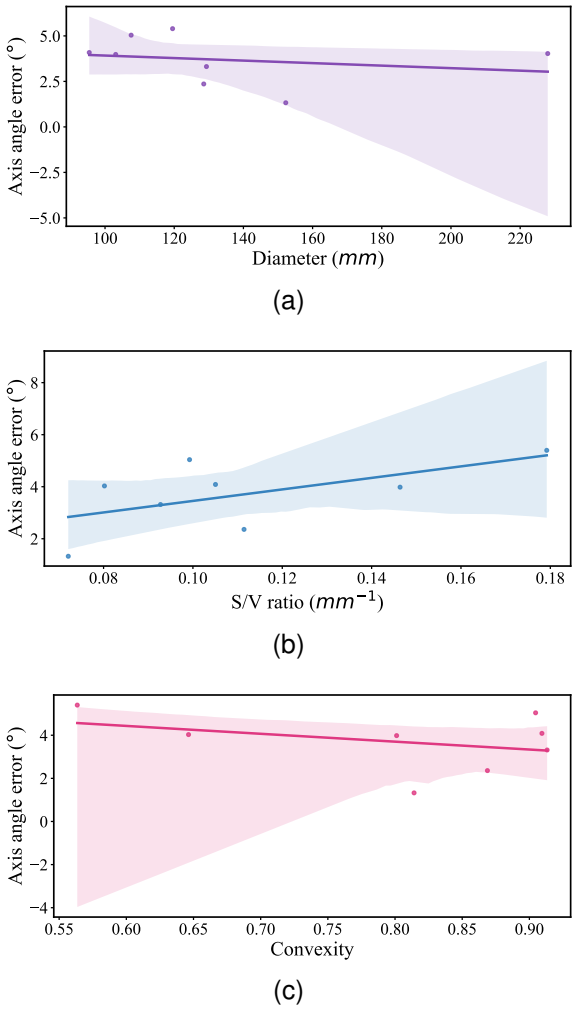

Fig. 12. Correlation between object size, shape, and algorithm accuracy. (a) Relationship between object diameter and axis-angle error. (b) Relationship between S/V ratio and axis-angle error. (c) Relationship between convexity and axis-angle error.

sampling issues of some normal frames. However, under the joint operation of both modules, the axis-angle error remains comparable to that of the basic framework. However, the ADD metric shows a significant improvement, demonstrating the effectiveness of the approach.

To demonstrate the broader applicability of the proposed method in the field of industrial automation, experiments were conducted to analyze the impact of object shape and size on pose tracking accuracy. Object size was evaluated using the same metric introduced in Section IV-C. Specifically, the diameter of the object, defined as the straight-line distance between the two farthest points in the 3D point cloud, was used as the measurement standard, with the unit in millimeters (mm). For object shape, complexity is typically considered. Two metrics were employed: surface-area-to-volume ratio (S/V ratio) and convexity. The S/V ratio is calculated by dividing the object's surface area by its volume. A higher S/V ratio indicates higher surface complexity. The unit for the S/V ratio is mm^{-1}, the inverse of length. Convexity is defined as the ratio between the object's actual volume and the volume of its convex hull, ranging from 0 to 1. A value closer to 0 indicates more

TABLE VII
CORRELATIONS BETWEEN THE ACCURACY OF THE OUTPUT 3D MODELS AND THE AXIS-ANGLE ERRORS ARE EXPLORED. FOR EACH OBJECT, THE NUMBER OF VERTICES IN THE 3D MODELS IS REDUCED FROM 100% TO 5%

Category	Correlation Coefficient	P-value
bear	-0.22	0.37
bird	-0.14	0.71
box	-0.43	0.07
dog	-0.02	0.95
doughnut	-0.22	0.39
rabbit	-0.18	0.51
statue	-0.41	0.11
train	-0.08	0.87
Average	-0.21	0.50

concave regions or holes in the object. Convexity serves as a dimensionless measure of the overall shape complexity. Using these three metrics, fitting and correlation analyses with axis-angle error were performed, as shown in Fig. 12. Each point in the figure represents a different object in the dataset, with the fitted line representing the results and the shaded background indicating the confidence interval during fitting. The calculated correlation coefficients between object diameter, S/V ratio, and convexity with axis-angle error were -0.22, 0.59, and -0.36, respectively. The results suggest that larger object diameters correspond to higher tracking accuracy, while objects with higher S/V ratios, indicating higher surface complexity, exhibit lower tracking accuracy. Additionally, objects with lower convexity, indicating higher overall shape complexity, also result in lower tracking accuracy. The large confidence intervals and relatively low correlation coefficients, however, indicate that object shape and size have a limited impact on tracking accuracy. In particular, the weak correlation between object size and accuracy can be attributed to the scaling during dataset capture and the use of 3D models for reference. Regarding object shape, the introduction of 3D models in the cross-space fusion model allows for effective handling, even for complex-shaped objects. This further validates the effectiveness and generalizability of the proposed method.

For most model-based object pose estimation works, the model accuracy is closely related to the algorithm performance. This relationship is particularly pronounced in neural network-based methods, which heavily rely on the matching between the input 3D model and 2D images. To investigate the relationship between 3D model accuracy and algorithmic error, an ablation experiment was designed. The 3D models in the dataset, captured by a 3D scanner, have model accuracies ranging from 30,000 to 50,000 vertices. Based on this, the 3D points of each model are simplified and 20 sets of comparisons are designed. The number of vertices is uniformly reduced from the original 100% down to 5%. The results of these comparisons are shown in Table VII. It can be observed that the correlation coefficients for all objects are negative, indicating that as the number of vertices decreases, the output accuracy of the system gradually decreases. However, the small absolute values of the correlation coefficients and the large p-values suggest a weak negative correlation between the number of object vertices and accuracy. This is attributable to the Cross-Spatial Fusion module, which ensures high accuracy in 3D point depth retrieval even with fewer vertices. However, when the number of object vertices is too low, the inability to generate densely packed projections leads to a rapid increase in error.

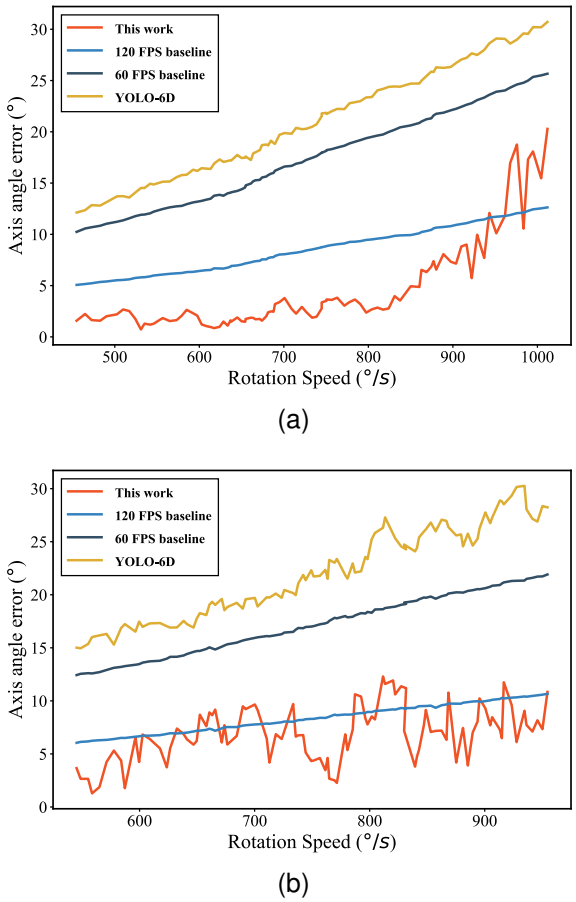

Fig. 13. Relationship between object rotation speed and algorithm accuracy. The plot includes results from this work, 60/120 FPS baselines, and YOLO-6D running under real-world conditions. (a) Box sequence. (b) Dog sequence.

This work primarily focuses on detecting high-speed rotating objects in industrial automation, and thus the impact of different speeds on algorithm accuracy is also an important aspect of the study. Additional speed comparison datasets are included using two objects with different shapes: the simple Box and the more complex Dog. The object speed is measured in terms of the angular velocity in degrees per second. The tested object speeds cover a range from approximately 450 degrees per second to over 1000 degrees per second across 93 sequences. Considering the 1000 FPS sampling frequency, even when the object's rotational speed increases, aliasing does not occur. In addition to the present work, comparisons are made with the 60 FPS and 120 FPS ground truth baselines mentioned in Section IV-D, as well as with the YOLO-6D model used in this study. The results are shown in Fig. 13. From the results, it is clear that the proposed method consistently maintains high accuracy levels. Even with a rotational speed of 1000 degrees per second, the tracking accuracy of our method is higher than the 120 FPS baseline but lower than the

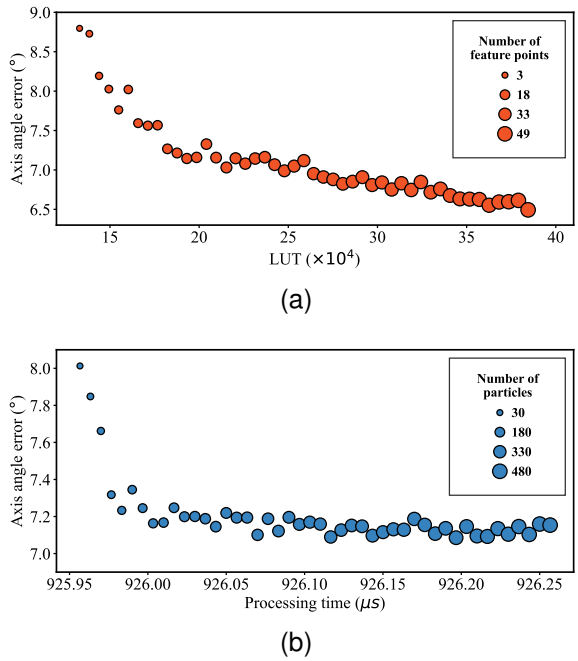

Fig. 14. Comparison of the number of feature points and particles in relation to algorithm accuracy, hardware resource usage, and speed. (a) Relationship between LUT (hardware resources)-axis angle error (accuracy)-number of feature points. (b) Relationship between processing time (speed)-axis angle error (accuracy)-number of particles.

60 FPS baseline and other works. This demonstrates that our method performs well even with high-speed rotations. For the "Dog" sequence, the fluctuations in the results are primarily due to the high variability in the reference results from the YOLO-6D work. The increase in tracking error with speed is mainly caused by the large difference between consecutive frames at high speeds, which affects feature point search and the accuracy of pose estimation in the particle filter.

The trade-off problem between algorithm accuracy, hardware resource usage, and speed is also important. The ablation experiments are conducted for the number of feature points and the number of particles, respectively. The number of feature points only impacts resource utilization, as the weight calculation is processed in parallel. The number of particles affects the number of pipeline repetitions in hardware, which in turn impacts the processing time. The experimental results are shown in Fig. 14. As the number of feature points and the number of particles increase, the axial angle error decreases while the LUT usage and processing time increase. When the number of feature points and particles reaches a certain level, the improvement in accuracy gradually becomes insignificant. Considering the limited hardware resources of the FPGA, the proposed system sets the number of feature points to 15 and the number of particles to 150.

V. Conclusion

In this paper, a high real-time performance object pose tracking method tailored for factory automation is presented, detailing both the algorithmic approach and hardware implementation. A feature points based FPGA-GPU hetero complementation architecture is proposed, effectively bypassing the delays associated with neural network computations. A cross-space fusion module leverages 3D model data as prior knowledge, enhancing the stability of feature points during object rotation and improving the accuracy of depth calculations of feature points with the integration of timing information. To support the high frame rate architecture, a dynamic particle filter is introduced. This method employs pipelining and parallel architecture to enhance tracking accuracy. A new mechanical device paired with a high frame rate camera has been used to produce an object pose measurement dataset specific to factory automation scenarios. Experimental evaluations conducted on both software and hardware platforms demonstrate that the proposed method achieves superior real-time performance with an accuracy comparable to similar software algorithms. A latency of less than 1 ms per frame is attained, corresponding to 1000 FPS, while utilizing 86.19% of the LUT resources.

This method still has several areas for improvement. On one hand, the Cross-Spatial Fusion module, which uses vertex stacking for depth measurement, is a simple and effective approach. However, for overly simple models (fewer than one thousand vertices), it fails to form a densely packed projection of indices. Therefore, techniques from computer graphics, such as rasterization, could be considered to perform the computation in a vectorized manner. Regarding the dynamic particle filter mentioned in this paper, while it currently provides an optimal solution given the context of the study, the number of particles is fixed due to hardware constraints. A hardware filter with a variable number of particles could be designed, allowing for space efficiency and improved accuracy. Additionally, a more efficient resampling method could be employed. As for the dataset, the current object capturing device is relatively rudimentary. The wooden frame limits its ability to capture larger objects and prevents high-speed rotation. A more stable capturing device could be built, incorporating multiple lighting conditions and backgrounds to enhance the dataset.

Future work will focus on improving the detection of challenging objects, such as those with weak textures or non-rigid structures. Enhancements to the robustness of the feature points tracking process will aim to prevent their occlusion by other objects within the factory environment. Additionally, the neural network used in this study can be further accelerated to reduce the key frame interval, and the algorithm can be optimized to consume fewer hardware resources, facilitating migration to other hardware platforms.

Acknowledgments

This work was supported by KAKENHI (21K11816).

References

[1] "Ieee standard glossary of software engineering terminology," *IEEE Std 610.12-1990*, pp. 1–84, 1990.
[2] L.-C. Chiu, T.-S. Chang, J.-Y. Chen, and N. Y.-C. Chang, "Fast sift design for real-time visual feature extraction," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3158–3167, 2013.
[3] J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari, "Real-time fully incremental scene understanding on mobile platforms," *IEEE Trans. Robot. Automat.*, vol. 3, no. 4, pp. 3402–3409, 2018.
[4] P. M. Athanas and A. L. Abbott, "Real-time image processing on a custom computing platform," *Computer*, vol. 28, no. 2, pp. 16–25, 1995.

[5] W. Qu and D. Schonfeld, "Real-time decentralized articulated motion analysis and object tracking from videos," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2129–2138, 2007.

[6] S. Du, K. Gu, and T. Ikenaga, "Subpixel displacement measurement at 784 fps: From algorithm to hardware system," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[7] F. Deng, J. Dong, X. Wang, Y. Fang, Y. Liu, Z. Yu, J. Liu, and F. Chen, "Design and implementation of a noncontact sleep monitoring system using infrared cameras and motion sensor," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 7, pp. 1555–1563, 2018.

[8] J. Li, B. Wang, S. Zhu, X. Cao, F. Zhong, W. Chen, T. Li, J. Gu, and X. Qin, "Bcot: A markerless high-precision 3d object tracking benchmark," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6697–6706.

[9] Y. Zhang, Y. Liu, Q. Wu, J. Zhou, X. Gong, and J. Wang, "Eanet: Edge-attention 6d pose estimation network for texture-less objects," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–13, 2022.

[10] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 17 868–17 879.

[11] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3003–3013.

[12] S. Li, H. Schieber, N. Corell, B. Egger, J. Kreimeier, and D. Roth, "Gbot: graph-based 3d object tracking for augmented reality-assisted assembly guidance," in *Proc. IEEE Conf. Virtual Reality and 3D User Interfaces*. IEEE, 2024, pp. 513–523.

[13] B. Qian, Y. Wang, R. Hong, and M. Wang, "Adaptive data-free quantization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 7960–7968.

[14] Z. Liu, Y. Wang, K. Han, S. Ma, and W. Gao, "Instance-aware dynamic neural network quantization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12 434–12 443.

[15] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable lookup table for neural network quantization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12 423–12 433.

[16] X. Chu, L. Li, and B. Zhang, "Make repvgg greater again: A quantization-aware approach," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 10, 2024, pp. 11 624–11 632.

[17] P. Wang, W. Chen, X. He, Q. Chen, Q. Liu, and J. Cheng, "Optimization-based post-training quantization with bit-split and stitching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2119–2135, 2022.

[18] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16 091–16 101.

[19] S. Gao, F. Huang, Y. Zhang, and H. Huang, "Disentangled differentiable network pruning," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 328–345.

[20] Z. Chen, J. Xiang, Y. Lu, Q. Xuan, Z. Wang, G. Chen, and X. Yang, "Rgp: Neural network pruning through regular graph with edges swapping," *IEEE Trans. Neural Networks Learn. Syst.*, 2023.

[21] P. Wu, Z. Wang, H. Li, and N. Zeng, "Kd-par: A knowledge distillation-based pedestrian attribute recognition model with multi-label mixed feature learning network," *Expert Syst. Appl.*, vol. 237, p. 121305, 2024.

[22] Y. Li, T. Hu, R. Fuchikami, and T. Ikenaga, "Global to multi-scale local architecture with hardwired cnn for 1-ms tomato defect detection," *IET Image Proc.*, 2024.

[23] T. Hu, R. Fuchikami, and T. Ikenaga, "Temporal prediction-based temporal iterative tracking and parallel motion estimation for 1 ms rotation-robust lk-based tracking system," *IEEE Trans. Instrum. Meas.*, 2023.

[24] T. Hu and T. Ikenaga, "Pixel selection and intensity directed symmetry for high frame rate and ultra-low delay matching system," *IEICE Trans. Inf. Syst.*, vol. 101, no. 5, pp. 1260–1269, 2018.

[25] Y. Yang, S. Du, and T. Ikenaga, "Search-free gridding and temporal local matching based observation for high frame rate and ultra-low delay slam system," in *Proc. IIAE Int. Conf. Intell. Syst. Image Process.*, 2018.

[26] P. Zhang, T. Hu, D. Luo, S. Du, and T. Ikenaga, "Highly-parallel hardwired deep convolutional neural network for 1-ms dual-hand tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 12, pp. 8192–8203, 2021.

[27] Z. Fan, Y. Zhu, Y. He, Q. Sun, H. Liu, and J. He, "Deep learning on monocular object pose detection and tracking: A comprehensive overview," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–40, 2022.

[28] Y. Zhu, M. Li, W. Yao, and C. Chen, "A review of 6d object pose estimation," in *Proc. Joint Int. Inf. Technol. Artif. Intell. Conf.*, vol. 10. IEEE, 2022, pp. 1647–1655.

[29] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *Int. J. Comput. Vision*, vol. 81, pp. 155–166, 2009.

[30] S. Gold, C.-P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness, "New algorithms for 2d and 3d point matching: Pose estimation and correspondence," *Adv. Neural Inf. Process. Syst.*, vol. 7, 1994.

[31] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Ieee, 1999, pp. 1150–1157.

[32] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," in *Proc. IEEE Int. Symp. Mixed Augment. Real.* IEEE, 2008, pp. 125–134.

[33] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *Int. J. Comput. Vision*, vol. 66, pp. 231–259, 2006.

[34] D. G. Lowe *et al.*, "Fitting parameterized three-dimensional models to images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 5, pp. 441–450, 1991.

[35] Y. Li, L. Gu, and T. Kanade, "Robustly aligning a shape model and its application to car alignment of unknown pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1860–1876, 2011.

[36] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, 1993.

[37] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* IEEE, 2010, pp. 1696–1703.

[38] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao, "Car make and model recognition using 3d curve alignment," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.* IEEE, 2014, pp. 285–292.

[39] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1521–1529.

[40] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Proc. Robot.: Sci. Syst.*, 2018.

[41] Y. Bukschat and M. Vetter, "Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach," *arXiv:2011.04307*, 2020.

[42] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16 611–16 621.

[43] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3828–3836.

[44] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 292–301.

[45] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4561–4570.

[46] X. Yu, Z. Zhuang, P. Koniusz, and H. Li, "6dof object pose estimation via differentiable proxy voting loss," *arXiv:2002.03923*, 2020.

[47] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 431–440.

[48] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 683–698.

[49] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in rgb," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 800–815.

[50] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7668–7677.

[51] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7678–7687.

[52] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1941–1950.

[53] T. Hodan, D. Barath, and J. Matas, "Epos: Estimating 6d pose of objects with symmetries," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 703–11 712.

[54] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6d object pose estimation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 108–125.

[55] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "inerf: Inverting neural radiance fields for pose estimation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2021, pp. 1323–1330.

[56] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou, "Onepose: One-shot object pose estimation without cad models," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6825–6834.

[57] X. He, J. Sun, Y. Wang, D. Huang, H. Bao, and X. Zhou, "Onepose++: Keypoint-free one-shot object pose estimation without cad models," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 35 103–35 115, 2022.

[58] L. Vacchetti, V. Lepetit, and P. Fua, "Stable real-time 3d tracking using online and offline information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1385–1391, 2004.

[59] C. Harris and C. Stennett, "Rapid-a video rate object tracker." in *BMVC*, 1990, pp. 1–6.

[60] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic, "Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds," *IEEE Trans. Visual. Comput. Graphics*, vol. 20, no. 1, pp. 99–110, 2013.

[61] X. Tian, X. Lin, F. Zhong, and X. Qin, "Large-displacement 3d object tracking with hybrid non-local optimization," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 627–643.

[62] H. Huang, F. Zhong, Y. Sun, and X. Qin, "An occlusion-aware edge-based method for monocular 3d object tracking using edge confidence," in *Computer Graphics Forum*, vol. 39, no. 7. Wiley Online Library, 2020, pp. 399–409.

[63] V. A. Prisacariu and I. D. Reid, "Pwp3d: Real-time segmentation and tracking of 3d objects," *Int. J. Comput. Vision*, vol. 98, pp. 335–354, 2012.

[64] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1328–1342, 2021.

[65] L. Wang, S. Yan, J. Zhen, Y. Liu, M. Zhang, G. Zhang, and X. Zhou, "Deep active contours for real-time 6-dof object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 14 034–14 044.

[66] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield, "Keypoint-based category-level object pose tracking from an rgb sequence with uncertainty estimation," in *Int. Conf. Robot. Autom.* IEEE, 2022, pp. 1258–1264.

[67] M. A. Al-yoonus and S. A. Al-Kazzaz, "Fpga-soc based object tracking algorithms: A literature review," *Al-Rafidain Engineering Journal (AREJ)*, vol. 28, no. 2, pp. 284–295, 2023.

[68] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros, "Parallel architecture for hierarchical optical flow estimation based on fpga," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 6, pp. 1058–1067, 2011.

[69] M. Tomasi, S. Pundlik, and G. Luo, "Fpga–dsp co-processing for feature tracking in smart video sensors," *J. Real-Time Image Process.*, vol. 11, pp. 751–767, 2016.

[70] G. K. Gultekin and A. Saranli, "An fpga based high performance optical flow hardware design for computer vision applications," *Microprocess Microsyst.*, vol. 37, no. 3, pp. 270–286, 2013.

[71] I. Ishii, T. Taniguchi, K. Yamamoto, and T. Takaki, "High-frame-rate optical flow system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 1, pp. 105–112, 2011.

[72] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari, and V. Murino, "A real-time versatile roadway path extraction and tracking on an fpga platform," *Comput. Vision Image Understanding*, vol. 114, no. 11, pp. 1164–1179, 2010.

[73] A. Jarrah, M. M. Jamali, and S. S. S. Hosseini, "Optimized fpga based implementation of particle filter for tracking applications," in *Proc. IEEE Natl. Aerosp. Electron. Conf.* IEEE, 2014, pp. 233–236.

[74] P. Engineer, R. Velmurugan, and S. Patkar, "Parameterizable fpga framework for particle filter based object tracking in video," in *Proc. IEEE Int. Conf. VLSI Des.* IEEE, 2015, pp. 35–40.

[75] U. Ali and M. B. Malik, "Hardware/software co-design of a real-time kernel based tracking system," *J. Syst. Archit.*, vol. 56, no. 8, pp. 317–326, 2010.

[76] P. Babu and E. Parthasarathy, "Fpga implementation of multi-dimensional kalman filter for object tracking and motion detection," *Eng Sci Technol Int J*, vol. 33, p. 101084, 2022.

[77] A. Kosuge, K. Yamamoto, Y. Akamine, and T. Oshima, "An soc-fpga-based iterative-closest-point accelerator enabling faster picking robots," *IEEE Trans. Ind. Electron.*, vol. 68, no. 4, pp. 3567–3576, 2020.

[78] P. Hobden, S. Srivastava, and E. Nurellari, "Fpga-based cnn for real-time uav tracking and detection," *Front. Space Technol.*, vol. 3, p. 878010, 2022.

[79] R. Konomura and K. Hori, "Fpga-based 6-dof pose estimation with a monocular camera using non co-planer marker and application on micro quadcopter," in *Proc. IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2016, pp. 4250–4257.

[80] W. Carballo-Hernández, M. Pelcat, and F. Berry, "Automatic cnn model partitioning for gpu/fpga-based embedded heterogeneous accelerators using geometric programming," *J. Signal Process. Syst.*, vol. 95, no. 10, pp. 1203–1218, 2023.

[81] X. Zhang, Y. Ma, J. Xiong, W.-M. W. Hwu, V. Kindratenko, and D. Chen, "Exploring hw/sw co-design for video analysis on cpu-fpga heterogeneous systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 6, pp. 1606–1619, 2021.

[82] X. Wang, K. Huang, and A. Knoll, "Performance optimisation of parallelized adas applications in fpga-gpu heterogeneous systems: A case study with lane detection," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 4, pp. 519–531, 2019.

[83] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem, "Robust visual tracking via consistent low-rank sparse learning," *Int. J. Comput. Vis.*, vol. 111, pp. 171–190, 2015.

[84] Y. Li, J. Zhu, and S. C. Hoi, "Reliable patch trackers: Robust visual tracking by exploiting reliable patches," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 353–361.

[85] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4335–4343.

[86] X. Qiu, W. Wu, and S. Wang, "Remaining useful life prediction of lithium-ion battery based on improved cuckoo search particle filter and a novel state of charge estimation method," *J. Power Sources*, vol. 450, p. 227700, 2020.

[87] Y.-H. Lin and X.-L. Jiao, "Adaptive kernel auxiliary particle filter method for degradation state estimation," *Reliab. Eng. Syst. Saf.*, vol. 211, p. 107562, 2021.

[88] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. IEEE Int. Conf. Comput. Vis.* Ieee, 2011, pp. 2564–2571.

[89] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[90] J. P. Grossman and W. J. Dally, "Point sample rendering," in *Proc. Eurographics Workshop on Rendering Techniques' 98, Vienna*. Springer, 1998, pp. 181–192.

[91] M. Schütz, B. Kerbl, and M. Wimmer, "Software rasterization of 2 billion points in real time," *Proc. ACM on Computer Graphics and Interactive Techniques*, vol. 5, no. 3, pp. 1–17, 2022.

[92] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7467–7477.

[93] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[94] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Proc. Asian Conf. Comput. Vis.* Springer, 2013, pp. 548–562.

[95] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.* IEEE, 2017, pp. 880–888.

[96] P.-C. Wu, Y.-Y. Lee, H.-Y. Tseng, H.-I. Ho, M.-H. Yang, and S.-Y. Chien, "[poster] a benchmark dataset for 6dof object pose tracking," in *Proc. IEEE Int. Symp. Mix. Augment. Real.* IEEE, 2017, pp. 186–191.

[97] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers, "A gauss-newton approach to real-time monocular multiple object tracking," *arxiv:1807.02087*, 2018.

[98] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," in *Int. Conf. Robot. Autom.* IEEE, 2020, pp. 9426–9432.

[99] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3343–3352.

[100] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *Int. Conf. Robot. Autom.* IEEE, 2017, pp. 1386–1383.

PLACE PHOTO HERE

Wei Wang received the B.E. degree in automation from Southeast University, Nanjing, China, in 2023. He is currently pursuing the M.E. degree with the Graduate School of Information, Production, and Systems of Waseda University, Japan. His current research interests include 3D computer vision and high-speed vision systems.

PLACE PHOTO HERE

Yuan Li received the B.E. degree in information engineering from Southeast University, China, in 2018, and the M.E. and Ph.D degrees in engineering of information, production and systems, in 2019 and 2024, from Waseda University, Japan. She is currently with the school of electrical and information engineering, Changzhou Institute of Technology, China. Her research focuses on hardware acceleration of computer vision algorithms.

PLACE PHOTO HERE

Tingting Hu received her B.E. degree in School of Automation from the Beijing Institute of Technology, China, in 2016, and her M.E. and Ph.D. degrees from the Graduate School of Information, Production, and Systems of Waseda University, Japan, in 2017 and 2023, respectively. She joined Panasonic Corporation, Japan, in 2018. Her current research interests are high frame rate and ultra-low delay vision systems.

PLACE PHOTO HERE

Ryuji Fuchikami received his B.Sc. degree in Kyushu Institute of Technology, Japan, in 1999. He joined Panasonic Corporation (Formerly Matsushita Electric Industrial Co., Ltd.), Japan, in 2000. He had been undertaking research on the LSI design of video codec. His current research interests are high-speed visual systems.

PLACE PHOTO HERE

Takeshi Ikenaga (Senior Member, IEEE) received his B.E. and M.E. degrees in electrical engineering and Ph.D degree in information computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively. He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he had been undertaking research on the design and test methodologies for high performance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI system design for image understanding processing. He is presently a professor in the integrated system field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are image and video processing systems, which covers video compression (e.g., VVC, SCC), video filter (e.g., super resolution, high-dynamic-range imaging), and video recognition (e.g., sports analysis, ultra-high speed and ultra-low delay vision system). He is a senior member of the Institute of Electrical and Electronics Engineers (IEEE), a member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) and the Information Processing Society of Japan (IPSJ).