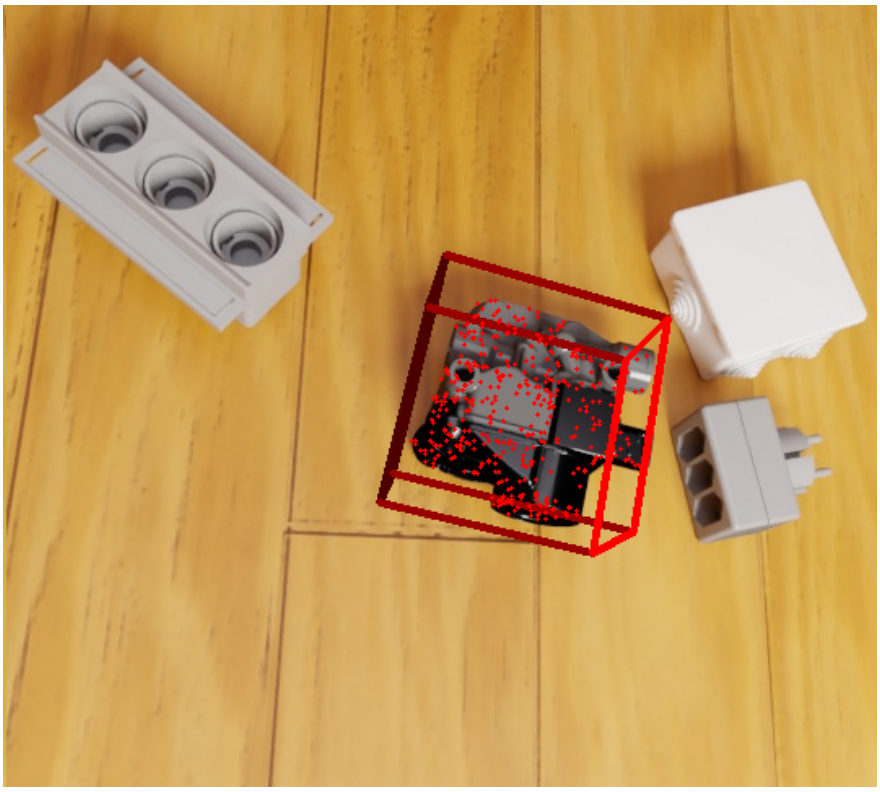
CHALMERS
UNIVERSITY OF TECHNOLOGY



Manipulation strategies of different objects applied to industrial settings

Identification of Objects and 6D Pose Estimation

Master's thesis in System, Control and Mechatronics

JUNJIE HU YU KANG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Manipulation strategies of different objects applied to industrial settings

Identification of Objects and 6D Pose Estimation

JUNJIE HU YU KANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
System and Control Division
PerceptionVolvoA
Chalmers University of Technology
Gothenburg, Sweden 2025

Manipulation strategies of different objects applied to industrial settings
Identification of Objects and 6D Pose Estimation
JUNJIE HU YU KANG

Supervisor: Atieh Hanna, Volvo Group Trucks Operations
Examiner: Karinne Ramirez-Amaro, Department of Electrical Engineering

Master's Thesis 2025
Department of Electrical Engineering
System and Control Division
PerceptionVolvoA
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 764438999

Cover: 6D pose estimation

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Manipulation strategies of different objects applied to industrial settings
Identification of Objects and 6D Pose Estimation
JUNJIE HU YU KANG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

6D pose estimation involves determining the three-dimensional position and orientation of an object, which is crucial for industrial automation tasks such as robot manipulation and assembly. Due to limitations such as occlusion, different lighting conditions, and complex geometric shapes, traditional methods based on deep learning often face difficulties in complex industrial environments. This article studies and compares two 6D attitude estimation methods. Firstly, the effectiveness of the YOLO-6D framework in factory environments was explored, which integrates contour-based learning and geometric constraints. Afterwards, the SAM-6D model was attempted for 6D pose estimation, which is a zero-sample method that utilizes the Segment Analysis model (SAM) for instance segmentation and hierarchical geometric inference. For the implementation of the two methods, Blenderproc2 was first used to generate a virtual dataset of factory parts, which is used to collect a set of images in the actual factory environment. Afterwards, the model is trained using a virtual dataset and tested using real images. The results indicate that although YOLO-6D was trained on a synthetic dataset, its performance on real images is poor due to the geometric complexity of the parts and limited model capacity. In contrast, SAM-6D exhibits excellent generalization ability in semantic alignment, appearance consistency, and geometric compatibility, achieving an accuracy of 97% in challenging simulation scenarios. This study emphasizes the importance of SAM-6D for 6D pose estimation in industrial applications where training data is scarce and object diversity is prevalent. The main contributions include optimizing the computational efficiency of SAM-6D by adjusting GPU-CPU allocation and verifying its robustness to traditional methods. These findings provide a foundation for deploying adaptive attitude estimation systems in dynamic industrial environments.

Keywords: 6D pose estimation, SAM-6D, YOLO-6D, Robot manipulation, industrial automation

Acknowledgements

On the occasion of completing this thesis, I express my sincerest gratitude to all the teachers, classmates, and friends who have provided me with support, guidance, and assistance during my graduate studies and thesis writing process.
Firstly, I would like to sincerely thank my mentor Professor Karinne Ramirez Amaro. Thanks to her careful guidance in academic research, from topic selection, experimental design to data analysis and paper revision, Professor Karinne Ramirez Amaro has always guided me with a rigorous academic attitude and profound professional knowledge. The teacher's profound insight into scientific problems, passion for academic research, and patient tolerance towards students have benefited me greatly and inspired me to constantly move forward on the path of scientific research.
Thank you to all members of the VOLVO Group, especially Atieh Hanna, for their technical support and valuable suggestions during the experiment. When encountering difficulties in a project, discussions with you always inspire new ideas. In addition, we would like to express our gratitude to Atieh Hanna for her guidance on experimental arrangements and equipment usage. Your assistance has provided important guarantees for the smooth completion of the paper.
Finally, I would like to express my high respect to all the experts and professors who participated in the paper review and defense! The end of my graduate career is the starting point of a new journey, and I will continue to deepen my knowledge with the gains from this experience.
This work was supported by the Vinnova project AIHURO (Intelligent human-robot collaboration). Junjie Hu Yu Kang, Gothenburg, MAY 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

SAM	Segment Anything Model
YOLO-6D	You Only Look Once for 6D Pose Estimation
PnP	Perspective-n-Point
RGB-D	Red Green Blue + Depth
ADD	Average Distance of Model Points
FPS	Frames Per Second
CNN	Convolutional Neural Network
ViT	Vision Transformer
SfM	Structure from Motion
CAD	Computer-Aided Design
PVNet	Pixel-wise Voting Network
DINOv2	Self-Distillation with No Labels v2
ROS2	Robot Operating System 2
MSE	Mean Squared Error
BCE	Binary Cross Entropy
CE	Cross Entropy
SVD	Singular Value Decomposition
BlenderProc2	Blender Procedural Dataset Generator v2
BOP	Benchmark for 6D Object Pose Estimation
YCB-V	YCB Video Dataset
T-LESS	Texture-Less Object Dataset
LM	LineMOD Dataset
LM-O	LineMOD-Occlusion Dataset
BB8	Bounding Box 8-vertex method
BOP	Benchmark for 6D Object Pose Estimation
IoU	Intersection over Union
ONNX	Open Neural Network Exchange
PBR	Physics-Based Rendering
SIMD	Single Instruction, Multiple Data
RMS	Root Mean Square
TF32	TensorFloat-32
MKL	Math Kernel Library
bg	background

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i, j	Indices for parallel edge groups
t	Time step index
x, y	Pixel coordinates in an image
m	Index for segmentation proposals
k	Index for keypoints in coordinate loss
c	Object category index
o	Object instance index

Sets

M	Set of high-confidence matching pairs
P_{Im}	Set of patches in the candidate image proposal
P_T	Set of patches in the template
$\mathcal{D}$	Input dataset of RGB-D images
C	Set of object classes

Parameters

fx, fy	Focal lengths of the camera
cx, cy	Camera principal point coordinates
K	Camera intrinsic matrix
τ	Threshold for segmentation or matching

δ_{vis}	Threshold for patch-level visibility
N	Number of keypoints
S	Grid size of feature map
M	Number of bounding boxes per grid cell
d_{th}	Distance threshold in confidence function
λ	Weighting coefficients for different loss terms

Variables

$\mathbf{p}_k^{pred}$	Predicted keypoint coordinate
$\mathbf{p}_k^{gt}$	Truth keypoint coordinate
$\hat{y}$	Predicted confidence
y	Truth confidence label
$\hat{y}_c$	Predicted class probability for class c
y_c	Truth label for class c
$m_{xy}, \hat{m}_{xy}$	Truth / predicted mask at pixel (x,y)
$\hat{l}_{ij}$	Predicted length of edge j in group i
L_{pc}	Coordinate (pose) loss
L_{conf}	Confidence loss
L_{id}	Classification loss
L_{mask}	Mask segmentation loss
L_{edge}	Edge consistency loss
$c(x)$	Geometry-aware confidence function
$s_{sem}, s_{appe}, s_{geo}$	Semantic, appearance, and geometric scores
A^c, A^f	Coarse and fine feature similarity matrices
r_{vis}	Visibility ratio
s	Overall similarity score
$\mathbf{R}, \mathbf{t}$	Rotation matrix and translation vector (6D pose)

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

6D attitude estimation is the cornerstone of spatial understanding in automated systems. The core of this technology is to calculate two different spatial parameters of the object: the 3D position coordinates (x, y, z) that accurately locate the object's position, and the rotation angles (roll, pitch, yaw) that define the object's direction [1]. These measurements establish a crucial connection between computer vision and real-world objects, and as more automation technologies are applied in the industry, this ability becomes increasingly important. The growing interest in this field stems from its widespread application across multiple departments [2]. In manufacturing factories, robots use these estimated values for precise component assembly. Augmented reality applications accurately overlay digital information onto physical objects. Autonomous vehicles use this technology to understand the location of roadside objects. Especially in factory automation, millimeter-level accuracy determines production quality, and reliable 6D attitude detection becomes indispensable.

The existing attitude estimation solutions are usually divided into two categories. The first type uses traditional pattern recognition techniques, such as the LINEMOD method, which compares objects with pre- stored templates using manually crafted features. The second type uses modern neural networks such as PoseCNN and YOLO6D, which can automatically learn features from data [3]. Although these methods work well under controlled laboratory conditions, their performance significantly decreases in real industrial environments. The factory environment brings various challenges that test the limits of these models. The lighting conditions between different workstations may vary greatly. Mechanical parts often appear partially hidden during the assembly process. Complex mechanical components containing multiple interconnected components can create visual confusion during assembly, causing confusion for recognition systems. These practical issues continue to drive research on more robust 6D pose recognition solutions.

Emerging technologies are exploring hybrid approaches to overcome current limitations. A promising direction is to combine object segmentation with zero sample learning strategies [4]. This combination allows the system to recognize objects without requiring a large amount of training data, while achieving high accuracy and stability. Our research focuses on comparing two different methods: the YOLO-6D framework, known for its speed, and the zero sample learning method SAM-6D, known for its flexibility. This study examined how these technologies handle indus-

trial components in challenging work environments, while evaluating the recognition stability and measurement accuracy of two models. This workflow consists of two main stages: first, isolate individual objects from RGB-D cameras, create a virtual 3D environment for the objects and cameras using a computer, and then use mathematical models to determine their exact positions and angles relative to the machine. This virtual environment consists of two parts, the first being a simple environment with only the target object and camera, followed by a complex environment that includes the target object, interfering objects, and camera. In the second stage, the improved model is tested under real factory conditions to verify its reliability in real-world environments, where different lighting conditions and surface textures make the task more complex.

1.1.1 YOLO-6D Model

The widespread adoption of RGB-D sensors in modern industrial environments has had a significant impact on 3D perception technology. These devices are typically integrated into robot work units and quality inspection systems, enabling the development of spatial analysis and coordinate regression methods using depth data. These methods, especially those utilizing point cloud processing, have introduced new computational frameworks for pose estimation tasks. However, the inherent dependence on depth information results in a significant amount of computation required for 6D pose recognition, which introduces significant limitations in practical implementation[5]. This technology limitation becomes particularly evident in cost-sensitive activities or environments lacking specialized hardware support, such as outdoor monitoring systems or mobile robot platforms, where the functionality of sensors limits the timeliness of 6D pose recognition.

Against the backdrop of the gradual advancement of artificial intelligence, the past few years have witnessed significant progress in visual computing capabilities. Driven by significantly enhanced computing power and available large-scale datasets, deep learning techniques have gradually matured, while catalyzing innovative methods that use RGB images as inputs. Representative architectures such as PoseCNN and PVNet demonstrate this trend of transformation, achieving measurable improvements in the accuracy of direct pose regression or the efficiency of keypoint voting [6]. These developments are consistent with the trend of more industries widely adopting computer vision to achieve automation solutions. However, the actual deployment of the algorithm reveals ongoing challenges in balancing algorithm performance and efficiency, most notably the inherent trade-off between repetition accuracy and speed in convolutional neural network design.

The project first adopts YOLO-6D as its framework, a streamlined architecture designed to overcome these operational limitations [7]. The design philosophy of the system emphasizes computational efficiency through two major innovations. The first innovation introduces a geometry-aware learning mechanism that incorporates contour data into the model. During model training, the method strategically uses contour information as important spatial cues, effectively guiding the network's feature extraction layers to prioritize geometric features. The practical implementation of the method involves parallel segmentation branches that generate object masks

simultaneously with the main detection output, enforcing detailed contour learning in early network layers. This architectural choice has proven to be particularly beneficial in cluttered industrial environments where partial occlusions and background noise are common.
The second innovation lies in geometric regularization through a specialized loss function. By formulating an edge-constrained optimization objective, the framework ensures that the 3D bounding box geometry is more strictly followed during the projection calculation. This mathematical formulation effectively minimizes shape distortions while keeping the computation tractable. Unlike traditional multi-stage pipelines that require sequential processing modules or high-data-volume methods that rely on synthetic training data generation, YOLO-6D adopts a unified prediction mechanism. The system directly outputs 2D projections of 3D bounding box vertices through a single forward pass, and then performs a Perspective-n-Point (PnP) algorithm for the final pose calculation. This process eliminates the traditional post-processing overhead while maintaining real-time performance.

1.1.2 SAM-6D Model

The accuracy of 6D pose estimation remains a key challenge in modern manufacturing systems. While YOLO-6D performs well in processing objects with basic geometric shapes such as cuboids and cylinders, its efficiency decreases when processing parts with complex contours such as turbine blades or automotive gearboxes [8]. This limitation stems from the difficulty of the framework itself in capturing complex surface geometries through standard bounding box representations. With the development of automated production lines, a large number of industrial parts on actual assembly lines present complex 3D geometries. This accuracy defect seriously limits the deployment potential of YOLO-6D in quality inspection and robotic grasping tasks.
To bridge this technological gap, the emerging zero sample 6D pose estimation model provides a promising alternative. This innovative method enables the system to perform geometric inference on new objects with minimal reference data while achieving very high 6D pose estimation accuracy. This ability is particularly valuable on flexible production lines where manufacturers frequently update product designs. For example, in the scenario of customizing automotive parts, the model can adapt to newly designed engine components within hours rather than weeks.
The rapid evolution of foundation models has accelerated progress in this domain. The Segment Anything Model (SAM) [9], developed through large-scale visual data training, has established new standards in image understanding. SAM's architecture combines three specialized modules: a high-capacity image encoder processing $640 * 480$ pixel inputs, a flexible prompt encoder accepting multiple input types (clicks, bounding boxes, text descriptions), and a lightweight mask decoder generating precise segmentation outputs [10].
However, direct application of SAM to industrial pose estimation encounters three primary obstacles [11]. First, the model tends to generate a large number of segmentation candidates, which complicates identifying target objects in cluttered workstations. Second, common factory conditions, such as appearance differences caused

by part materials or reflection and refraction of light, often lead to mismatches between segmented regions and actual 3D models. Third, traditional 6D pose estimation methods that require step-by-step iterations greatly increase computation time, with the prototype system taking more than 10 seconds per frame, slowing down the efficiency of real-time robot operations.
The SAM-6D framework used in this study addresses these challenges through systematic innovations. The solution implements a three-stage filtering process. The model first uses the material recognition function of DINOv2 for initial semantic filtering and removes some irrelevant segmentation proposals. After that, the texture of the part in multiple camera views is compared in the subsequent appearance matching process. The final geometry verification uses fast IoU calculation between the point cloud and the CAD model. Unlike template-based methods that require a lot of database maintenance, SAM-6D redefines the process of pose estimation as dynamic point cloud alignment. Actual tests using factory parts show that the success rate of pose estimation is 97% in the environment of a single target object and 91% in the environment with interference. The model basically meets the strict industrial real-time requirements while maintaining accuracy.

1.1.3 Project Aim

This research program is supported by the Volvo Group Quality and Engineering department and focuses on developing a reliable 6D pose recognition system for the automotive parts assembly process in smart factories. The project is designed for assembly lines in factory environments, where robot arms need to grasp multiple different parts with extremely high accuracy. The core output of the system includes three key elements: accurate 6D pose coordinates for robot path planning, oriented bounding boxes for collision avoidance, and dense point cloud models for quality verification.
During the system development process, a CAD model of a part on the assembly line was first used to create a virtual dataset about the part. At the same time, in order to verify the validity of the virtual dataset, a small amount of data from the real environment was collected. The collection process uses the Intel RealSense D435i camera mounted on the robotic arm to capture RGB-D images of the object at different positions and orientations, and uses a script to record information such as the object's rotation matrix. The reason for using this method to train the model is that the project faces significant data scarcity challenges. Although industrial cameras can capture many frames of data per hour, manual 6D pose annotation requires annotating the rotation and translation information of each frame of the image at the same time, which makes the cost of creating a real-world dataset too high.
To overcome this limitation, BlenderProc2 was used to generate synthetic data. This open source tool is able to batch render more than 1,000 synthetic images for each component category, simulating a variety of material textures and environmental conditions, such as on wooden boards, metal planes, grass, etc. The virtual dataset retains the geometry of the target component, while adjusting the pose of the object and the lighting in the environment through Blender's physics engine. Model verifi-

cation includes two stages of testing. First, the model's performance was tested on 40 synthetic test scenarios, and then the model was validated using data collected in real-world environments.
The project's primary technical challenges emerged from three aspects:

- Geometric complexity: Components that require 6D pose estimation have non-uniform surfaces, which poses a challenge to standard bounding box detection.
- Environmental dynamics: There are lighting changes in the production workshop, such as refraction, reflection, etc., which can interfere with 6D pose recognition.
- Hardware limitations: Using the SAM-6D model for 6D pose recognition of target objects requires a significant amount of computation, and the GPU memory used in the project has limitations on the computation, which affects the effectiveness of the model.

Key contributions of this industrial-academic collaboration include:

- Developed a pipeline based on BlenderProc2 to generate 3000 synthetic RGB-D images for selected components, combining realistic material properties and environmental interference.
- Verified that the YOLO-6D model cannot accurately estimate the 6D pose of parts with complex geometric structures and surface materials in complex industrial environments.
- Verified SAM-6D's 97% attitude estimation accuracy for complex parts in simple environments (only target object) and 91% attitude estimation accuracy in complex environments(target object and interfering object), despite its high computational requirements for hardware.
- Using dynamic adjustment calculations to optimize hardware acceleration attitude estimation, although this may reduce the accuracy of the model.

1.1.4 Proposed Approach Summary

This industry-university-research project focuses on the challenges of 6D gesture recognition in industrial scenarios and proposes a solution based on the collaboration of synthetic and real data. In order to meet the part grasping requirements of Volvo's production line, a complete technical path including virtual data generation, real-time detection framework and computational optimization was developed. The main innovations are in the following three aspects. A synthetic dataset containing a variety of surface materials and lighting conditions was constructed using BlenderProc2, which effectively alleviated the difficulty of obtaining real data. The feasibility of YOLO-6D in complex part gesture recognition and the high-precision recognition characteristics of SAM-6D for complex geometric shapes were verified. The hardware requirements for improving the SAM-6D reasoning process were explored. Actual tests show that the system can meet the grasping accuracy requirements of at least 91% of production line parts, but the calculation method still needs to be optimized in complex scenarios. The research results provide data generation standards and model selection basis for the rapid deployment of industrial vision systems, and have engineering practical value for improving the perception ability of intelligent manufacturing equipment.

1.2 Related works

1.2.1 YOLO-6D Model

The widespread adoption of RGB-D sensors has promoted the emergence of deep learning-based methods for 6D pose estimation. With the advancement of computer vision technology, depth-based pose recognition has become a core technology in robotics grasping and augmented reality applications. Taking agricultural applications as a typical case, Hwang et al. [12] developed a deep learning-based tool specifically designed for 6D pose recognition of fruits. In smart agriculture scenarios, accurate fruit pose recognition plays a decisive role in ensuring the operational precision of automated harvesting systems. Their experimental results indicate that this approach maintains high accuracy and demonstrates remarkable stability when operating in complex environments containing multiple targets and dynamic interference factors. In another important study, Wang et al. [13] achieved effective estimation of 6D pose information from single or multiple images through the application of Deep Convolutional Neural Networks (Deep CNN). This method overcomes traditional visual algorithms' dependency on multi-view images. Their experimental data particularly highlight the method's significant advancement in angular measurement accuracy for 6D pose determination.

The continuous advancement of deep learning technology has significantly propelled methodological innovations in 6D pose estimation. As a fundamental task in computer vision, the improvement of pose estimation accuracy directly impacts the performance of downstream applications. Early representative networks like PoseCNN [14][15] adopted a two-stage processing architecture: first employing fully convolutional networks for semantic segmentation, then utilizing Hough voting mechanisms for pose calculation. However, this architecture exhibits noticeable limitations in practical applications. Such stepwise processing tends to accumulate errors in industrial scenarios. Specifically, the model's stability degrades substantially when processing scenes with multiple object occlusions. Moreover, the system shows susceptibility to rotational estimation errors. To address these challenges, Rad and Lepetit [16] introduced the BB8 model featuring a dual-stage processing pipeline. The initial stage leverages convolutional neural networks to predict 2D projections of 3D bounding box vertices, followed by PnP algorithm application for final 6D pose resolution. The incorporation of geometric constraints effectively mitigates limitations of purely data-driven approaches. Comparative experiments confirm this model maintains high recognition accuracy even under partial object overlap conditions. In a parallel development direction, Peng et al. [17] proposed the PVNet model based on vector field prediction. The core innovation lies in predicting directional vectors from image pixels to object keypoints, combined with RANSAC algorithm optimization. This probabilistic modeling approach enhances tolerance to noisy data. This design significantly enhances model robustness in scenarios involving object occlusion and image truncation. Nevertheless, it should be noted that the improved accuracy comes at the cost of reduced computational efficiency.

Improved YOLO-based architectures demonstrate better balance between computational efficiency and recognition accuracy in 6D pose estimation. This balanced

characteristic endows them with unique advantages in real-time monitoring scenarios. Maji et al. [18] developed a multi-object pose estimation framework through enhancements to YOLOX. The technical implementation involves two main phases: first establishing 3D-2D keypoint correspondences, then applying the PnP algorithm for pose calculation. Notably, this framework achieves both object detection and pose estimation through single RGB image processing without requiring additional post-processing steps. Such end-to-end processing reduces system complexity. However, detailed analysis reveals certain limitations in geometric consistency constraints and shape verification within this methodology [19].

The recently proposed YOLO-6D model addresses existing deficiencies through innovative technical modifications. By incorporating contour prediction branches based on privileged learning algorithm [19][20], the enhanced architecture enables simultaneous learning of object contour information during training, thereby effectively improving feature representation robustness [7]. This multi-task learning strategy fully utilizes spatial structural information in images. Furthermore, the introduction of edge constraint loss function enhances shape consistency by imposing penalties on length deviations between predicted 3D bounding box parallel edges [7]. The explicit expression of geometric constraints improves the model's perception of object morphology. Experimental validation on LINEMOD dataset demonstrates significant performance improvements: 4.09% increase in 2D projection accuracy and 11.72% enhancement in ADD metric, while maintaining real-time processing capability at 71 FPS [7]. These metrics confirm the model's successful inheritance of the efficiency advantages of the YOLO structure while surpassing previous methods in accuracy.

Despite the improved YOLO-6D model's demonstrated robustness and accuracy, two critical challenges require further investigation. These limitations may affect algorithm deployment effectiveness in real industrial environments. First, current validation experiments primarily involve objects with simple geometric configurations, whereas industrial production scenarios often contain components with complex geometries and surface characteristics. For instance, robotic end-effectors typically exhibit asymmetric structures and reflective surfaces. Second, the model's adaptability across different RGB-D camera systems remains unverified. This study specifically focuses on evaluating the model's capability to maintain recognition accuracy and operational robustness when deployed with diverse camera systems in actual production line environments.

1.2.2 Segment Anything Model

Recent advancements in general visual segmentation have witnessed remarkable breakthroughs, with the Segment Anything Model (SAM) emerging as a landmark framework due to its exceptional zero-shot generalization capabilities and high accuracy [21]. This breakthrough fundamentally reshapes the paradigm of instance segmentation in open-world scenarios. The model implements instance segmentation in complex scenarios through a multimodal prompting system incorporating various input types including points, bounding boxes, and textual descriptions. Such multimodal interaction significantly enhances human-computer collaboration efficiency in annotation tasks. Its fundamental innovation resides in constructing an efficient

encoder-decoder architecture that bridges visual features and semantic understanding. This architectural design ensures effective knowledge transfer between low-level visual patterns and high-level semantics. The remarkable success of SAM has stimulated extensive cross-disciplinary adaptation studies. A notable application comes from medical imaging research, where Chen et al. [22] developed the BA-SAM architecture as an extension of the original SAM framework. Due to the scarcity of annotated data and high precision requirements in this field, medical image analysis particularly benefits from SAM's generalization ability. Comprehensive evaluations across three public medical datasets (BUSI,REFUGE2, andISIC-17) demonstrate BA-SAM's superior performance in processing biomedical images. However, critical analysis reveals persistent limitations including partial semantic information loss and relatively high computational demands. These shortcomings pose challenges for clinical applications requiring real-time processing.

To overcome the semantic deficiency inherent in SAM-based models, Shahraki et al. [23] introduced the SAMNet++ architecture as a hybrid 3D segmentation solution. Three-dimensional segmentation tasks demand richer spatial feature representation compared to 2D cases. This approach synergistically combines SAM's generalization capacity with PointNet++'s local feature extraction capabilities, enabling automatic semantic label assignment. The geometric prior knowledge from PointNet++ compensates for SAM's limitations in spatial reasoning. Parallel developments include the CNOS and PerSAM frameworks, which address semantic limitations through distinct technical pathways: the former employs feature similarity filtering mechanisms, while the latter leverages cross-image contrastive learning strategies [4]. These diversified approaches reflect the research community's multifaceted understanding of semantic representation.

Computational efficiency optimization represents another crucial research direction for SAM improvements. Deployment constraints in edge computing scenarios drive the demand for lightweight variants. Batista et al. [24] implemented the FastSAM model for aquatic biological monitoring, specifically targeting size measurement of Pacific oysters in tank environments. Aquaculture management requires accurate monitoring of organisms with limited computing resources. The technical innovation involves replacing the original Vision Transformer (ViT) architecture with lightweight convolutional networks, achieving substantial acceleration in inference speed without compromising segmentation quality. This substitution strategy maintains accuracy while reducing computational complexity. In a complementary approach, Wang et al. [25] applied the MobileSAM framework to construction site management, focusing on tower crane segmentation and tracking. Construction site safety monitoring demands both real-time response and reliable target recognition. Through parameter decoupling and knowledge distillation techniques, this method successfully reduces encoder complexity while maintaining functionality. Feature extraction preserves the basic feature representation ability of the original model. Experimental comparisons confirm MobileSAM's tenfold processing speed advantage over the original SAM model under identical operating conditions, alongside complete target segmentation capability. Such efficiency improvements significantly enhance practical deployment feasibility.

1.2.3 Pose Estimation of objects

Unsupervised 6D pose estimation methodologies have primarily evolved through two distinct technical paradigms: geometric inference rooted in template matching and deep learning approaches centered on feature alignment. These programs address the critical challenge of reducing dependency on annotated training data in industrial applications. Early implementations in this domain, exemplified by the Gen6D model developed by Liu et al. [26], pioneered template-free estimation by constructing a multi-view template database containing diverse object perspectives. Multi-view representation circumvents single-view ambiguity but introduces storage complexity. However, the model's precision was constrained by discrete sampling intervals during template generation. Discrete sampling may miss critical viewpoints, leading to pose estimation drift. Addressing these limitations, Manuelli et al.[4] introduced the MegaPose framework with an innovative rendering-comparison architecture, incorporating a hybrid strategy that combines coarse-grained classification with neural optimization processes. This hierarchical approach balances computational load and precision requirements. This methodology achieved performance breakthroughs in Benchmark for 6D Object Pose Estimation (BOP) benchmark evaluations.

Feature matching-based approaches emphasize the implicit acquisition of geometric relationships through data-driven learning. Such methods adapt well to environmental variations through learned feature invariance. Sun et al. [27] devised the OnePose framework that bypasses CAD model dependencies through a novel implementation. This independence from CAD models significantly enhances practical applicability in field deployments. The system utilizes Structure-from-Motion (SfM) techniques to build 3D feature representations, establishing robust 2D-3D correspondences that enable category-agnostic pose estimation without category-specific network training. SfM's self-supervised nature aligns with unsupervised learning principles. In a parallel development, Chen et al. [28] created the ZeroPose model incorporating a Discovery-Oriented Registration (DOR) mechanism. DOR's incremental learning capability supports dynamic industrial environments. This architecture demonstrates extensibility to novel objects without retraining requirements, effectively eliminating time-intensive training phases while enhancing computational throughput. Rapid deployment becomes feasible for production line reconfiguration. Nevertheless, current implementations still encounter persistent challenges including elevated computational complexity and optimization instability in partial occlusion scenarios. Occlusion-induced feature ambiguity remains a key obstacle for real-world adoption.

The recently introduced SAM-6D architecture presents significant advancements over conventional models through novel architectural integration, cross-modal fusion addresses the semantic-geometric disconnection in traditional approaches. By synergistically combining SAM's zero-shot segmentation capabilities with a multi-modal matching-scoring mechanism, the framework enhances instance localization robustness [4]. Zero-shot learning reduces annotation burdens in industrial component recognition. Furthermore, the model incorporates a background token mechanism that streamlines partial matching computations, thereby optimizing processing efficiency. Attention mechanism pruning improves real-time performance without sacrificing accuracy. However, critical questions remain regarding its performance

consistency when handling geometrically complex industrial components and the computational resources required for practical deployment. This research initiative specifically investigates SAM-6D's application for 6D pose estimation of intricate mechanical parts in factory environments, utilizing RTX4070 GPU hardware. Mainstream GPU selection ensures findings' relevance to common industrial setups. A primary objective involves assessing computational feasibility on existing GPU configurations and exploring potential optimization strategies for SAM-6D implementations. Energy-efficient deployment is crucial for sustainable manufacturing practices.

2
Theory

2.1 YOLO6D model

The YOLO-6D architecture implemented in this study consists of two consecutive processing stages, which represent a typical paradigm in modern object pose estimation frameworks. As an extension of the classical YOLO detection framework, this architecture integrates geometric reasoning capabilities while maintaining real-time processing characteristics.
During the model training phase, the two-dimensional projection of the object's three-dimensional boundary geometry is segmented from the input image. This critical preprocessing step establishes the foundation for subsequent geometric analysis by extracting precise contour information through deep neural networks. The segmentation process utilizes pixel-level annotations to learn the mapping relationship between image space and 3D geometry space, which is particularly important for handling complex occlusion scenarios.
In the subsequent stage, the Perspective-n-point(PnP) calculation method is used to derive the six degrees of freedom spatial configuration parameters through geometric pose estimation. This classical computer vision algorithm has been widely adopted in pose estimation tasks due to its mathematical stability and computational efficiency. By solving the correspondence between 2D image points and 3D model points, the PnP algorithm effectively bridges the gap between image observations and physical spatial configurations.

2.1.1 Network Architecture

YOLO6D takes RGB images as input, which aligns with mainstream computer vision systems due to the sensor characteristics of conventional cameras and the intuitiveness of color information representation. The network architecture comprises three core components: a feature-sharing module, a contour prediction branch, and a bounding box estimation branch, as illustrated in Figure 2.1. This modular design philosophy ensures synergistic collaboration between different functional units while maintaining computational efficiency.
Specifically, the feature-sharing module consists of 14 layers, including 10 convolutional and 4 pooling layers that generate multi-scale feature representations through input data processing. The 14-layer depth configuration balances model expressiveness and computational tractability, allowing effective feature abstraction without excessive parameter overhead. These hierarchical operators progressively extract visual patterns from primitive edges to complex semantic features.

The contour prediction branch adopts an encoder-decoder semantic segmentation framework, where the feature-sharing module functions as the encoder for visual feature embedding, while the contour prediction branch serves as the decoder to reconstruct pixel-wise object masks through multi-level feature fusion. This architecture benefits from preserving spatial details through skip connections while maintaining global contextual awareness.

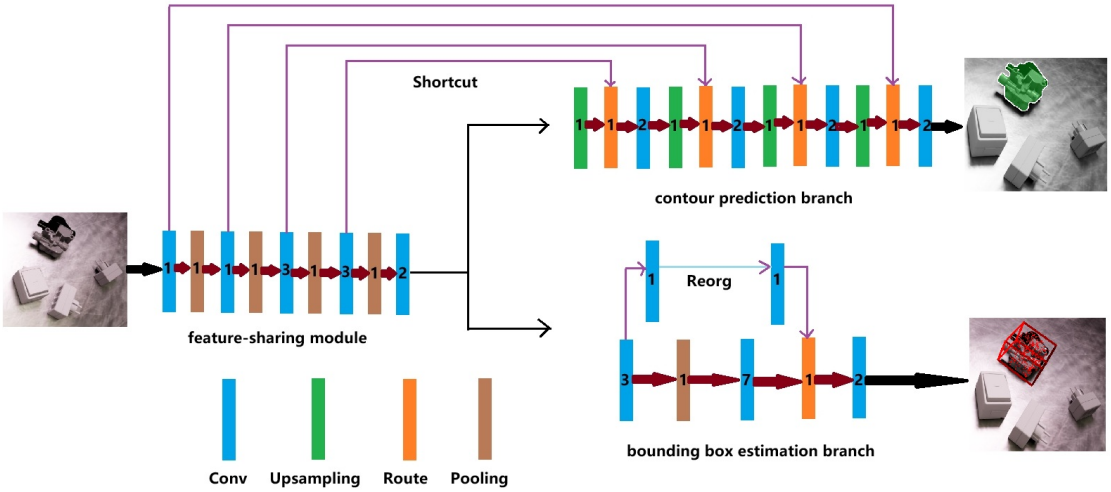


Figure 2.1: YOLO-6D network structure

In the contour prediction subnetwork, the encoder-decoder architecture incorporates symmetric feature transformation paths to ensure geometric consistency during the reconstruction process. The decoder consists of 4 upsampling modules, 7 convolutional operation units, and 4 cross-layer feature concatenation modules, forming a progressive refinement pipeline. Each upsampling stage employs bilinear interpolation to enhance feature map spatial resolution, a computationally efficient approach that maintains smooth gradient propagation. The channel-wise concatenation operation with encoder features enables effective multi-scale feature fusion, combining high-resolution shallow features with semantically rich deep features.
The fused features undergo local context modeling through 3×3 convolutional kernels that capture neighborhood relationships, followed by channel dimension compression via 1×1 convolutional kernels for computational optimization. The network terminates with a Sigmoid activation function for binary mask prediction, while other hidden layers utilize Leaky-ReLU activation functions for nonlinear transformation, striking a balance between gradient stability and representation capability.
In the bounding box estimation pipeline, feature maps undergo transformation through a sequence of fully convolutional layers that preserve spatial correspondence. The network discretizes the input image into $S * S$ regular grid cells via spatial grid partitioning, a design that enables parallel processing of local regions. This module outputs a feature tensor with dimensions $S * S * D$, where each spatial cell associates with a composite vector containing geometric parameters and semantic information. This vectorization strategy allows simultaneous prediction of multiple geometric attributes within unified feature representation.

The composite vector comprises 2D coordinate predictions for 9 keypoints (corresponding to projections of 8 vertices and the centroid of a 3D bounding box), object category probability distribution, and instance existence confidence. The selection of 9 keypoints provides sufficient geometric constraints while avoiding excessive computational complexity. Assuming M candidate bounding boxes are predicted per grid cell, the vector dimension can be calculated as:

$$D = (N * 2 + C + 1) * M \tag{2.1}$$

where N denotes the number of keypoints and C represents the total object categories. This parametric formulation ensures scalability across different detection scenarios.
To achieve geometric registration between 3D bounding boxes and target models, the method employs a predefined topological structure using 9 keypoints - specifically selecting 2D projections of 8 vertices and the centroid from 3D bounding boxes. This geometric prior knowledge effectively constrains the solution space of pose estimation problems. For 6D pose estimation tasks, a geometry-aware confidence evaluation mechanism is designed, replacing conventional IoU metrics with a confidence function $c(x)$ defined as:

$$c(x) = e^{1-\frac{D(x)}{d_{th}}} \tag{2.2}$$

The confidence function $c(x)$ establishes probabilistic evaluation through 2D Euclidean distance $D(x)$ between predicted keypoints and truth annotations. This exponential formulation emphasizes prediction accuracy near ground truth positions while maintaining differentiable properties for gradient-based optimization. d_{th} represents a distance threshold used to normalize the prediction error $D(x)$. A small d_{th} makes the confidence more sensitive to small distance changes, while a large d_{th} makes the function more tolerant to larger prediction errors.

2.1.2 Loss function

Edge constraint loss is a geometric prior loss function introduced in YOLO-6D, which plays a crucial role in ensuring geometric consistency for 6D pose estimation tasks. In computer vision applications, maintaining geometric validity under perspective projection remains a fundamental challenge. The purpose of introducing edge constraint loss is to constrain the projected shape of the predicted 3D bounding box on the image plane to conform to the geometric rules of perspective projection, thereby improving prediction accuracy. The formula is defined as:

$$L_{\text{edge}} = \sum_{i=1}^{3} \sum_{j=1}^{4} \left(\frac{l_{ij}}{l_i^*} - 1 \right)^2, \quad l_i^* = \frac{1}{4} \sum_{j=1}^{4} l_{ij} \tag{2.3}$$

Here, l_{ij} denotes the length of the predicted 3D bounding box on the j th edge in the i th parallel edge group, while l_i^* calculates the average length of the i th parallel edge group. The design rationale for this loss function operates at two levels. First, perspective projection causes parallel 3D edges to appear non-parallel in 2D images, but their relative length proportions should follow projective geometry constraints.

By minimizing deviations from the mean edge length, this loss effectively reduces shape distortion. Second, geometric constraints help the model prioritize structural consistency over superficial features, enhancing robustness against occlusion and background clutter - common challenges in real-world pose estimation scenarios.
The total loss function of YOLO-6D integrates multiple objectives through weighted multi-task learning, reflecting the complexity of simultaneous detection and pose estimation. This comprehensive formulation balances different aspects of the prediction task:

$$L_{total} = \lambda_{\text{edge}} L_{\text{edge}} + \lambda_{\text{pc}} L_{\text{pc}} + \lambda_{\text{conf}} L_{\text{conf}} + \lambda_{\text{id}} L_{\text{id}} + \lambda_{\text{mask}} L_{\text{mask}} \tag{2.4}$$

The first is called coordinate loss L_{pc}, which addresses the fundamental requirement of position accuracy. As 2D keypoints correspond to projections of 3D bounding box corners, their precise prediction directly impacts pose estimation quality. The mean square error (MSE) formulation emphasizes Euclidean distance minimization:

$$L_{\text{pc}} = \frac{1}{N} \sum_{k=1}^{N} \left\| \mathbf{p}_k^{\text{pred}} - \mathbf{p}_k^{\text{gt}} \right\|^2 \tag{2.5}$$

Among them, N is the total number of key points, which is usually 9 in YOLO-6D. $\mathbf{p}_k^{\text{pred}}$ is the position of the k-th keypoint predicted by the model on the 2D image (usually a 2D coordinate). $\mathbf{p}_k^{\text{gt}}$ is the true coordinate of the k-th keypoint.
Secondly, there is a confidence loss L_{conf}, which aims to distinguish between the grid containing the target (high confidence) and the background grid (low confidence). The method for calculating this loss is weighted binary cross entropy (BCE). For the grid containing the target, the weight is $\lambda_{\text{conf}} = 5$ and the background grid is $\lambda_{\text{conf}} = 0.1$. The formula is:

$$L_{\text{conf}} = -\lambda_{\text{conf}} \left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}) \right] \tag{2.6}$$

Among them, y represents whether the target is included, and $\hat{y}$ represents the confidence level of the prediction.
Thirdly, there is the classification loss L_{id}. The goal of this loss is to ensure proper object identification, a prerequisite for category-specific pose estimation. The cross entropy (CE) formulation is standard for multi-class recognition:

$$L_{\text{id}} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{2.7}$$

Among them, C enumerates object categories, y_c denotes ground-truth labels, and $\hat{y}_c$ represents class probabilities.
Finally, there is the mask loss L_{mask}, which aims at leveraging privileged segmentation information during training to enhance spatial understanding. The pixel-wise binary cross entropy operates across the entire image resolution:

$$L_{\text{mask}} = -\frac{1}{HW} \sum_{x=1}^{H} \sum_{y=1}^{W} \left[m_{xy} \log(\hat{m}_{xy}) + (1 - m_{xy}) \log(1 - \hat{m}_{xy}) \right] \tag{2.8}$$

Among them, $H * W$ is the image resolution, m_{xy} is the true mask value, and $\hat{m}_{xy}$ is the predicted value.

2.2 SAM-6D model

The SAM-6D model, developed for zero-shot 6D object pose estimation, represents a significant advancement in handling unseen objects without requiring prior training data. In modern computer vision systems, the ability to estimate 6D poses for arbitrary objects using RGB-D inputs has broad applications in robotics, augmented reality, and industrial automation. As depicted in Figure 2.2, the model architecture decomposes the 6D pose estimation task into two sequential processing stages through dedicated subnetworks, following a widely adopted modular design philosophy in pose estimation research.

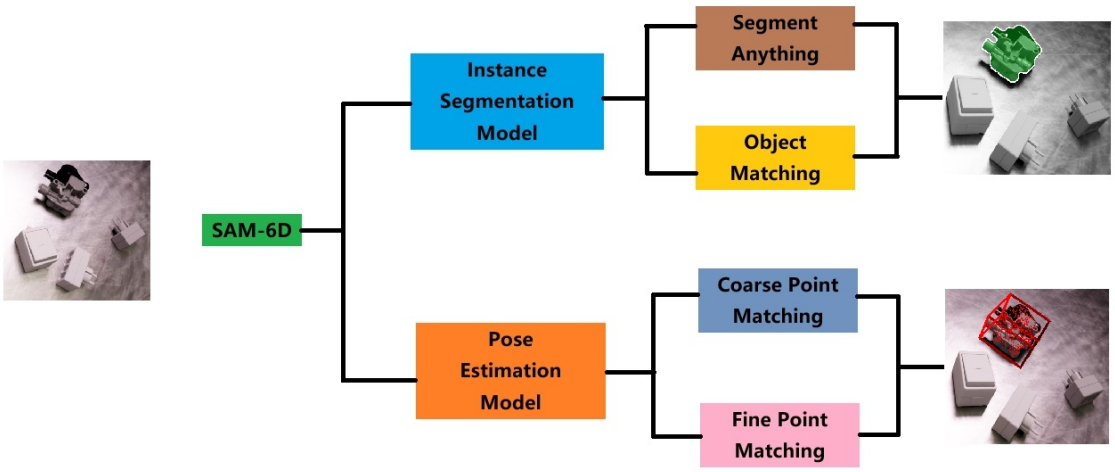


Figure 2.2: SAM-6D structure

The initial stage utilizes an Instance Segmentation Model (ISM) to extract object instances from RGB-D imagery. This segmentation phase serves as the critical first step, isolating individual objects from complex backgrounds to enable focused pose analysis. The RGB-D data combination provides complementary information - color features from RGB channels enhance recognition, while depth data aids in spatial reasoning. Following successful segmentation, the Pose Estimation Model (PEM) independently calculates 6D pose parameters for each segmented instance through geometric analysis and feature matching. This systematic division of segmentation and pose estimation functions creates a cascaded processing pipeline that enables the model's zero-shot generalization capability, particularly valuable for scenarios requiring rapid adaptation to new objects.
The two-stage architecture offers multiple operational advantages. First, the separation allows specialized optimization of segmentation and pose estimation components separately. Second, it prevents error propagation between different processing objectives through modular isolation. Third, this design pattern aligns with human visual perception mechanisms that first locate objects before analyzing their spatial relationships. The following sections detail the implementation and operational principles of these two constituent models, with particular emphasis on their synergistic collaboration in achieving zero-shot performance.

2.2.1 Instance Segmentation Model

2.2.1.1 SAM model

The Instance Segmentation Model (ISM) is employed to extract individual instances of a target object O, a critical preprocessing step for subsequent pose estimation. In modern computer vision pipelines, accurate instance segmentation directly impacts downstream task performance by ensuring object localization precision. In this process, the model takes an RGB image I as input and utilizes the Segment Anything Model (SAM), which is a foundational vision model renowned for its zero-shot segmentation capabilities – to generate a set of candidate proposals M. For each proposal $m \in M$, ISM computes a similarity score s through multi-criteria evaluation, aiming to evaluate the degree of correspondence between the proposal and the target object in terms of semantics, appearance, and geometric alignment. This comprehensive scoring mechanism ensures robustness against partial occlusions or lighting variations. The best-matching proposal is identified by comparing the score against a predefined threshold τ, implementing a simple yet effective selection strategy:

$$s = \text{Sim}(M, O), \quad \text{if } s > \tau, \text{ then } M \text{ is considered a match to } O \tag{2.9}$$

In SAM, instance segmentation is guided by prompts P, which may include points, bounding boxes, text, or masks from the input image. This flexible prompting mechanism enables adaptable interaction with diverse input modalities. The model consists of three main components: the Image Encoder f_{img} for visual feature extraction, the Prompt Encoder f_{prompt} for processing user guidance signals, and the Mask Decoder f_{mask} for generating pixel-wise predictions. The segmentation process is formalized as follows:

$$M, C = f_{\text{mask}}(f_{\text{img}}(Z), f_{\text{prompt}}(P)) \tag{2.10}$$

where M denotes the set of predicted candidate masks and C represents the corresponding confidence scores, reflecting prediction certainty at the pixel level.
In this study, all potential proposals are generated using a uniformly sampled 2D grid over the input image. This grid-based initialization ensures comprehensive spatial coverage of the input image, reducing the risk of missing small or partially visible objects. These proposals are subsequently filtered based on their confidence scores C, retaining only the high-confidence masks through thresholding operations. This filtering strategy improves the precision and robustness of the final instance segmentation results by eliminating low-quality candidates, thereby reducing computational overhead in downstream processing stages.

2.2.1.2 Object matching score

The object matching score systematically evaluates candidate proposals through multi-modal alignment analysis, a critical process for ensuring pose estimation accuracy in zero-shot scenarios. This tripartite scoring mechanism combines complementary criteria to address diverse matching challenges encountered in real-world

perception tasks. Match and score each proposal in terms of semantics, appearance, and geometry. By comparing the local details block by block, ensure that the candidate proposals are highly consistent with the target object in terms of visual features, thereby mitigating errors from partial occlusions or lighting variations.

2.2.1.2.1 Appearance Matching Firstly, appearance matching is used to measure the consistency between candidate proposals and target objects in surface details such as texture and color, distinguishing objects with similar semantics but different appearances. This granular comparison operates at the patch level to capture fine-grained visual patterns. In this process, the template T_{best} that is most similar to the proposal is first determined through semantic matching. Afterwards, the ViT model of DINOv2, renowned for its discriminative feature representation capabilities, is used to extract the features $\boldsymbol{f}^{\text{patch}}_{\mathcal{I}_m,j}$, which is the visual feature vector of the j patch in the candidate image, and the features $\boldsymbol{f}^{\text{patch}}_{\mathcal{T}_{\text{best}},i}$, which is the visual feature vector of the i patch in the template image T_{best}. Next, calculate the maximum similarity between each image block j in the proposal and all image blocks in the template, implementing a robust many-to-one correspondence strategy:

$$\max_{i=1,\ldots,N^{\text{patch}}_{\mathcal{T}_{\text{best}}}} \left\langle \boldsymbol{f}^{\text{patch}}_{\mathcal{I}_m,j}, \boldsymbol{f}^{\text{patch}}_{\mathcal{T}_{\text{best}},i} \right\rangle \tag{2.11}$$

The maximum similarity of all proposed image blocks is averaged to obtain the appearance score, ensuring a comprehensive evaluation across the entire candidate region:

$$s_{\text{appe}} = \frac{1}{N^{\text{patch}}_{\mathcal{I}_m}} \sum_{j=1}^{N^{\text{patch}}_{\mathcal{I}_m}} \max_{i=1,\ldots,N^{\text{patch}}_{\mathcal{T}_{\text{best}}}} \left\langle \boldsymbol{f}^{\text{patch}}_{\mathcal{I}_m,j}, \boldsymbol{f}^{\text{patch}}_{\mathcal{T}_{\text{best}},i} \right\rangle \tag{2.12}$$

Among them, $N^{\text{patch}}_{\mathcal{I}_m}$ is the total number of patches divided in the candidate image, used to average the matching results of all patches.

2.2.1.2.2 Geometric Matching Next is geometric matching, which aims to evaluate the matching degree between candidate proposals and target objects in terms of shape, size, and spatial position, and exclude erroneous proposals caused by occlusion or segmentation errors. This spatial reasoning process consists of three coordinated steps. The first step performs rough pose estimation to establish spatial correspondence. Generate a rough initial pose using the rotation parameters of the best matching template T_{best} and the proposed average depth position. Based on this pose, project the target object onto the image plane to obtain the projected bounding box B_o , creating a geometric reference for subsequent analysis. The second part calculates the Intersection over Union (IoU) between the projected bounding box B_o and the candidate proposed bounding box B_m , a standard metric for spatial overlap evaluation:

$$s_{\text{geo}} = \frac{\mathcal{B}_o \cap \mathcal{B}_m}{\mathcal{B}_o \cup \mathcal{B}_m} \tag{2.13}$$

Step three performs visibility ratio correction to account for occluded regions, enhancing geometric validity. Determine whether each image block of template r_{vis} is visible in the candidate proposals. If the similarity between a block and any block in the proposal exceeds the threshold, it is considered visible, implementing adaptive visibility assessment:

$$r_{\text{vis}} = \frac{1}{N_{\mathcal{T}_{\text{best}}}^{\text{patch}}} \sum_{i=1}^{N_{\mathcal{T}_{\text{best}}}^{\text{patch}}} \mathbb{I}\left(\max_{j=1,\ldots,N_{\mathcal{I}_m}^{\text{patch}}} \frac{\left\langle \boldsymbol{f}_{\mathcal{I}_m,j}^{\text{patch}}, \boldsymbol{f}_{\mathcal{T}_{\text{best}},i}^{\text{patch}} \right\rangle}{\|\boldsymbol{f}_{\mathcal{I}_m,j}^{\text{patch}}\| \cdot \|\boldsymbol{f}_{\mathcal{T}_{\text{best}},i}^{\text{patch}}\|} \geq \delta_{\text{vis}} \right) \tag{2.14}$$

The final geometric score combines spatial overlap with visibility awareness through multiplicative fusion:

$$s_{\text{geo}} \leftarrow r_{\text{vis}} \cdot s_{\text{geo}} \tag{2.15}$$

2.2.1.2.3 Semantic Matching Finally, semantic matching establishes category-level consistency. Firstly, generate a target object template by synthesizing multi-view representations. Sample N_τ poses of the target object O, render a template image $T_1, T_2, T_3, ..., T_n$ for each pose, and crop the background to preserve the object area, creating comprehensive viewpoint coverage. Afterwards, leverage pre-trained DINOv2 ViT model's generalization power – the candidate proposed RGB image blocks I_m are input to extract global class embeddings $\boldsymbol{f}_{\mathcal{I}_m}^{\text{class}} \in \mathbb{R}^C$ (where C is the feature dimension). Next, for each template T_i , calculate the cosine similarity between its class embedding and the candidate proposed class embedding, and select the top K templates with the highest similarity, implementing selective attention to most relevant prototypes. The final calculation formula ensures discriminative semantic alignment:

$$s_{\text{sem}} = \frac{1}{K} \sum_{k=1}^{K} \max_{i=1,\ldots,N_{\mathcal{T}}} \frac{\left\langle \boldsymbol{f}_{\mathcal{I}_m}^{\text{class}}, \boldsymbol{f}_{\mathcal{T}_i}^{\text{class}} \right\rangle}{\|\boldsymbol{f}_{\mathcal{I}_m}^{\text{class}}\| \cdot \|\boldsymbol{f}_{\mathcal{T}_i}^{\text{class}}\|} \tag{2.16}$$

2.2.1.2.4 Score Fusion The final matching score integrates semantic, appearance, and geometric information through adaptive weighting, balancing different evidence types according to visibility conditions. The calculation formula implements visibility-aware normalization:

$$s_m = \frac{s_{\text{sem}} + s_{\text{appe}} + r_{\text{vis}} \cdot s_{\text{geo}}}{2 + r_{\text{vis}}} \tag{2.17}$$

Among them, s_{sem} is the semantic matching score, s_{appe} is the appearance matching score, r_{vis} is the adaptive visibility assessment, and s_{geo} is the geometric matching score.

2.2.2 Pose Estimation Model

The pose estimation model addresses the 6D pose recovery challenge through establishing a correspondence framework, aligning partial observations with complete

object models. This approach models the 6D pose estimation problem as a partial-to-partial point cloud matching problem, that is, by establishing the correspondence between the target object model point cloud and the observation point cloud, the optimal rigid body transformation is solved. Such formulation effectively handles occlusions and incomplete sensor data common in real-world scenarios. The three-stage pipeline (feature extraction, rough matching, and fine matching) progressively refines pose accuracy while balancing computational efficiency. The structure of pose estimation model is shown in Figure 2.3, illustrating the cascaded processing flow from raw inputs to final pose output.

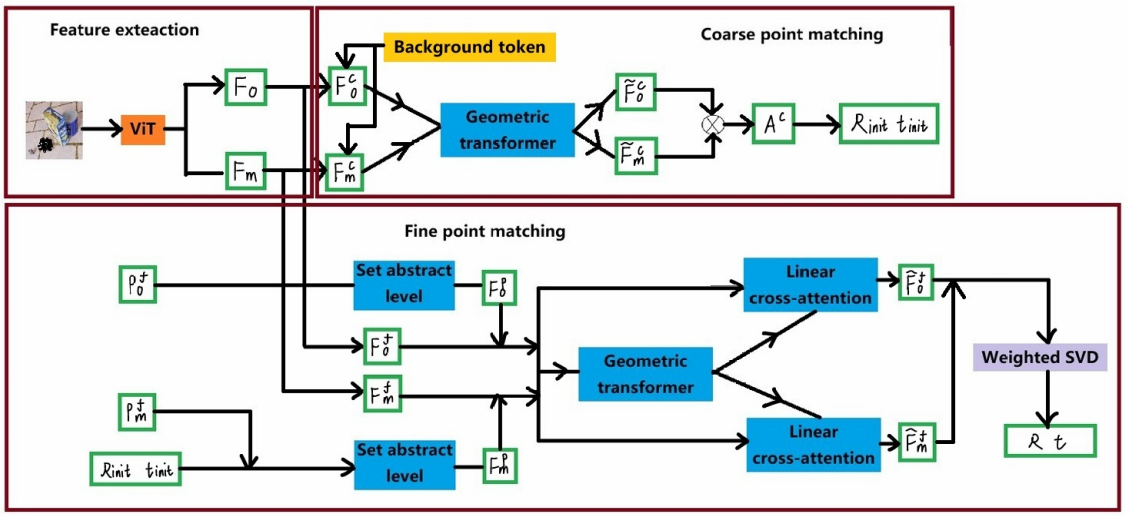


Figure 2.3: Structure of pose estimation model

2.2.2.1 Feature Extraction

The feature extraction stage bridges 2D visual patterns and 3D geometric structures. In this process, the inputs are the observation point cloud of the proposal m, denoted as $P_m \in \mathbb{R}^{N_m \times 3}$ (masked and coordinate transformed) and the target object point cloud, denoted as $P_o \in \mathbb{R}^{N_o \times 3}$ from the depth map of the candidate proposal. The ViT model of DINOv2 - pre-trained on large-scale visual datasets - is used to extract discriminative features of the candidate proposal image blocks. Through bilinear interpolation and coordinate mapping, these 2D visual features are projected to 3D coordinates to obtain the observation point features $F_m \in \mathbb{R}^{N_m \times C}$. The same operation is performed on the RGB image of the target object template to aggregate multi-view features, compensating for viewpoint variations and generating comprehensive target point features $F_o \in \mathbb{R}^{N_o \times C}$. The final output result is the dense features F_m and F_o of the observation point cloud and the target point cloud, establishing a shared embedding space for subsequent matching.

2.2.2.2 Coarse Matching

The coarse matching phase implements efficient hypothesis generation through sparse correspondence analysis. The purpose of coarse matching is to quickly estimate the initial pose R_{init} and t_{init} through sparse point sampling, providing a reliable starting

point for refinement. Firstly, sample $N_m^c = N_o^c = 196$(m means proposal, o means object, c means coarse matching) sparse points from P_m and P_o respectively to obtain P_m^c and P_o^c , balancing computational load and spatial coverage. Afterwards, add learnable background(bg) tokens f_m^{bg}, f_o^{bg} for each point to represent non-matching points, enabling the model to handle outlier points effectively. The 3-layer geometric Transformer models intra-point-set relationships (self-attention) and cross-set interactions (cross-attention), producing enhanced features $\tilde{\boldsymbol{F}}_m^c, \tilde{\boldsymbol{F}}_o^c$ with contextual awareness. The calculation formula for the soft allocation matrix implements dual normalization to suppress ambiguous matches:

$$\mathcal{A}^c = \text{Softmax}_{\text{row}}\left(\frac{\tilde{\boldsymbol{F}}_m^c(\tilde{\boldsymbol{F}}_o^c)^\top}{\tau}\right) \odot \text{Softmax}_{\text{col}}\left(\frac{\tilde{\boldsymbol{F}}_m^c(\tilde{\boldsymbol{F}}_o^c)^\top}{\tau}\right) \tag{2.18}$$

Here, τ serves as the temperature coefficient, which controls the sharpness of the matching distribution - lower values sharpen the probability mass on dominant correspondences. Finally, based on the $\mathcal{A}^c$ sampling points, generate candidate poses and select the hypothesis of minimum geometric error as the initial pose, ensuring rotational and translational coherence.

2.2.2.3 Fine Matching

The fine matching stage refines initial estimates through dense geometric reasoning. The purpose of precise matching is to optimize dense point matching using the initial pose and output accurate pose R, t . Firstly, the dense point cloud is sampled from proposal and object, which is P_m^f and P_o^f to achieve fine matching. Multi-scale set abstraction layers extract hierarchical position encoding F_m^p, F_o^p (p means position) , capturing both local geometric details and global structural patterns. Afterwards, strategic sampling selects a sparse subset from the dense point cloud, focusing computational resources on regions with high matching potential. Geometric Transformer interactions model complex spatial relationships between remaining points. The dense soft allocation matrix $\mathcal{A}^f$ identifies reliable correspondences through confidence thresholding $\tilde{\boldsymbol{F}}_m^f, \tilde{\boldsymbol{F}}_o^f$. High-confidence matching pairs are fed into a weighted SVD solver:

$$\min_{R,t} \sum_{(i,j)\in\mathcal{M}} w_{ij}|R\boldsymbol{p}_i + t - \boldsymbol{q}_j|^2 \tag{2.19}$$

where weights w_{ij} reflects matching confidence, The $\boldsymbol{p}_i$ reflects i point in the observation point cloud and the $\boldsymbol{q}_j$ reflects j point in the template point cloud that matches it. The coordinates of the observation point mapped to the target coordinate system through rigid body transformation can be calculated by $R\boldsymbol{p}_i + t$. This closed-form solution ensures optimal rigid transformation under measurement uncertainties, finalizing the 6D pose estimation.

3
Methods

3.1 Create simulation dataset

In this project, Blenderproc2 and BOP Toolkit are used to create a dataset for object. The generation of synthetic data plays a crucial role in computer vision research, particularly when real-world data collection faces challenges such as equipment costs, environmental constraints, and annotation difficulties. BlenderProc2 is an open-source Python library based on Blender, specifically designed for generating high-quality synthetic datasets, primarily used for training and testing machine learning (especially computer vision, robotics, deep learning). It combines Blender's powerful 3D rendering capabilities, supporting physical properties such as object collision, lighting simulation, and material reflection. This capability allows researchers to generate photorealistic images with precise annotations. It can randomize object layout, camera perspective, lighting conditions, and background, enhancing data diversity. At the same time, it also has powerful automatic scripting capabilities, which can output annotation information including RGB images, depth maps, instance segmentation masks, 6D pose (position+rotation) of objects, bounding boxes, and more. These comprehensive outputs form the foundation for subsequent model training and evaluation.

BOP (Benchmark for 6D Object Pose Estimation) is a standardized benchmark for evaluating 6D object pose estimation algorithms, and BOP Toolkit is its official toolkit for data processing, result evaluation, and visualization. The establishment of this benchmark has significantly promoted the development of 6D pose estimation technology. This toolkit supports loading and preprocessing mainstream BOP datasets such as LM, LM-O, T-LESS, YCB-V, etc. It can also calculate 6D attitude error metrics, classification accuracy, and average accuracy. At the same time, it can also generate error heatmaps and pose comparison maps to assist algorithm tuning, while maintaining a unified algorithm output format to ensure comparability of results from different methods. This standardized evaluation process helps researchers objectively compare algorithm performance.

The collaborative use of BlenderProc2 and BOP Toolkit enables the complete process from data generation to evaluation loop. This closed-loop workflow ensures the quality and usability of synthetic data. The generated dataset mainly includes the following information: object segmentation mask, depth map, RGB image, object bounding box, object pose, etc. These multi-modal data provide rich input features for deep learning models. The process of creating a virtual dataset is shown in Figure 3.1.

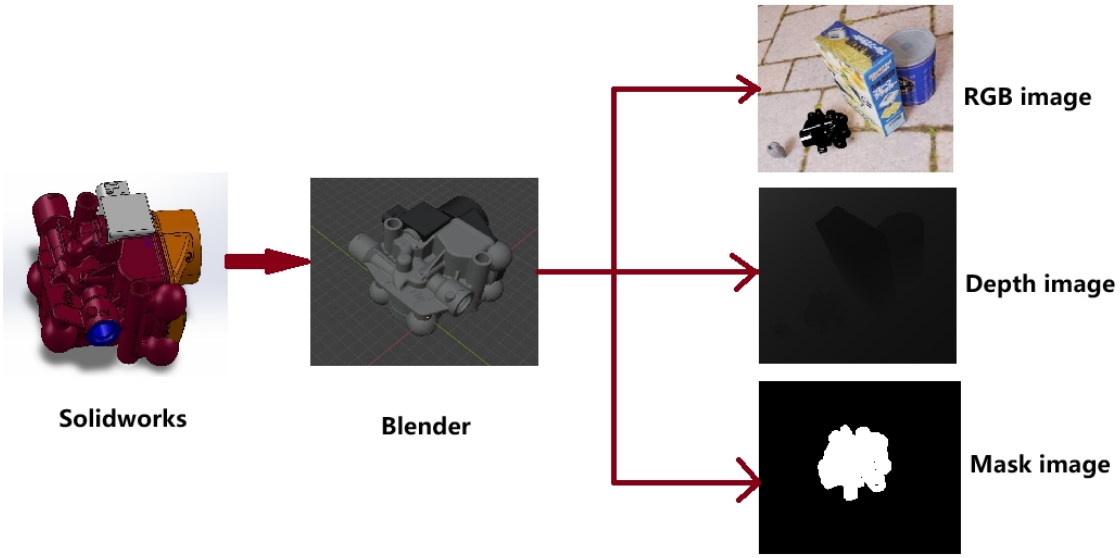


Figure 3.1: The process of generating virtual datasets

The process of creating a dataset is as follows:

1. Create a 3D modeling file of the parts to be assembled in Solidworks software and convert it to .obj format. Solidworks is chosen for its precision in mechanical design. The CAD design of the parts is shown in Figure 3.2.
2. Open the part using Blender software and adjust its size to 1000 times (Blender use meter as unit, however, Solidworks use millimeter), with the orientation set to face up. This scaling operation ensures dimensional consistency between virtual and real environments.
3. Set the origin of the model to the bottom right corner of the model (consistent with the origin in the actual environment). Proper origin alignment facilitates subsequent pose calculations.
4. Print the material color onto the node because .ply format files cannot save the material color of the model. This process preserves visual appearance characteristics.
5. The model needs to be saved as a .ply file. And import the .ply format model file into Blenderproc2. The PLY format is widely supported in 3D processing frameworks.
6. The lighting and camera information in Blender do not need to be changed as Blenderproc2 does not use this information. The rendering parameters will be controlled programmatically.
7. Use meshlab software to obtain the size information of the model and store it in the model _info. json file. This metadata is essential for physical simulation.
8. Write camera data to the camera.json file. Camera configuration directly affects the imaging characteristics.
9. Download the background image to the root directory of Blenderproc2 using the commands in Appendix 1.1. In the process of generating the dataset, Blenderproc2 randomly selects images as backgrounds, however, these images do not contain the actual working background of the robotic arm. Diverse

backgrounds improve model robustness.

10. Use the commands in Appendix 1.3 to generate the dataset. This automated process ensures batch data production.

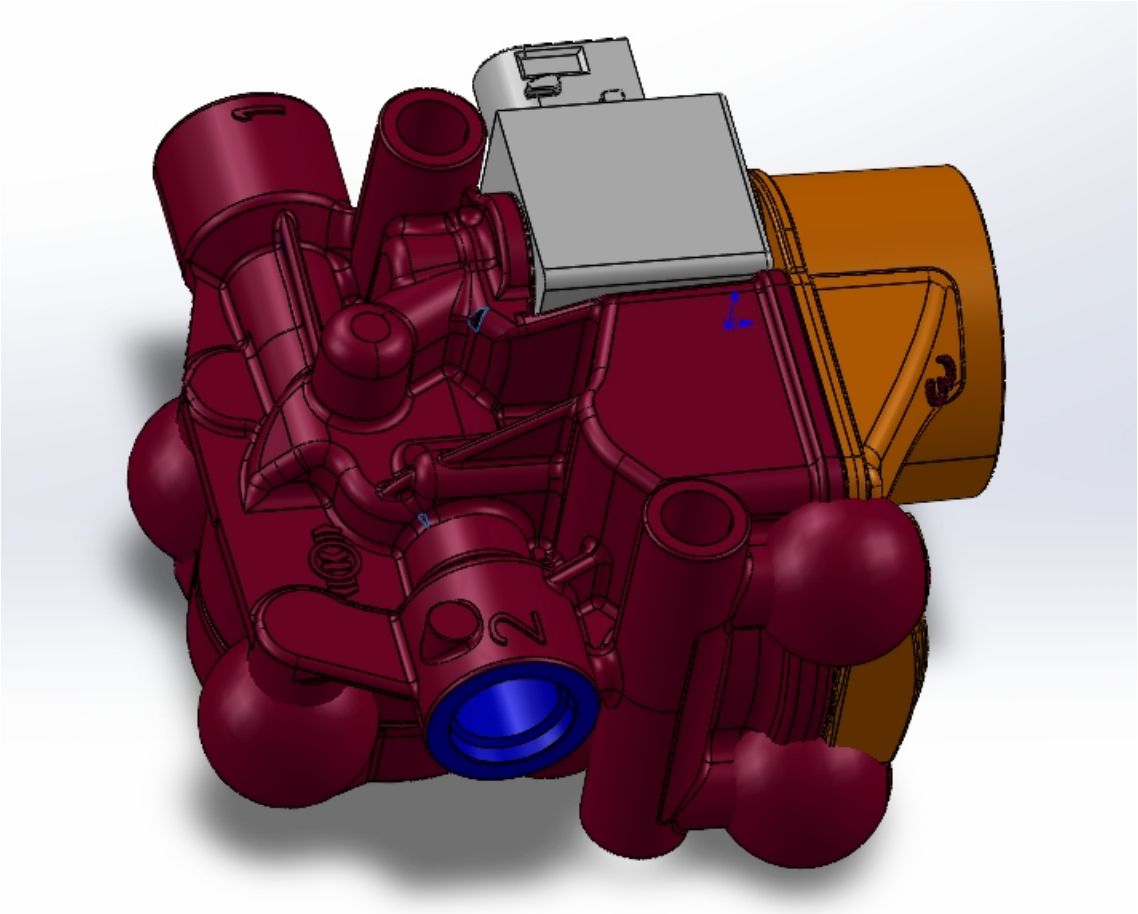


Figure 3.2: CAD design drawings of parts

The parameters of the camera are obtained through camera calibration. Camera calibration is a fundamental step in computer vision applications. Through calibration, the width, height, and intrinsic parameter matrix of the camera can be obtained. The internal parameter matrix of the camera is as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

This matrix encapsulates the essential geometric properties of the imaging system. The camera used in this project is Intel Realsense D435i, and the camera parameters are shown in Table 2.1. This depth camera is widely adopted in robotics research.

Name	Explain	Numerical value
cx	principal point in x	663
cy	principal point in y	367
depth_scale	Depth scaling factor	1.0
fx	focal length in x	907
fy	focal length in y	906
height	Image height	720
width	picture width	1280

Table 3.1: Camera information

During the generation of the dataset, Blenderproc2 can automatically load some models from other datasets as interference into the dataset to simulate possible occlusion, lighting changes, and other situations. This data augmentation strategy enhances model generalization capability. The datasets used in this project are the Tess, YCBV, and Tyol datasets. The models in these three datasets have appropriate sizes and simple geometric shapes, which can reduce the impact of interference on recognition accuracy. Extract these three datasets to the root directory of Blenderproc2. The code in Appendix 1.2 controls the number and extraction methods of models extracted from other datasets. This flexible control mechanism allows customized data complexity adjustment.

3.2 Test dataset collection

We collect data from the test dataset in the actual environment. This physical data acquisition phase is essential for validating the effectiveness of synthetic datasets and ensuring model generalization in real-world applications. The collection process is designed to maintain consistency with simulation conditions while addressing practical constraints. The specific workflow is as follows:

1. Install the camera on the robotic arm. The camera mounting position is carefully calibrated to ensure optimal field of view coverage while avoiding mechanical interference during arm movement.
2. Place the parts on the desktop and fix one corner of the camera as the origin. This spatial alignment establishes a unified coordinate system between the physical setup and virtual environment, which is crucial for subsequent pose estimation accuracy.
3. Select 10 positions separately and input the motion information of the 10 positions into the robotic arm. These pre-programmed positions are strategically distributed within the workspace to maximize spatial sampling diversity.
4. Control the rotation of the robotic arm and collect receipts. Use pre written scripts to synchronize data collection with arm movements, ensuring temporal consistency between motion execution and data recording.
5. Rotate the part 90 degrees around the origin and repeat step 4. This quarter rotation strategy regularly captures multi view observations while maintaining geometric constraints.

We collected a total of 40 sets of data through this rigorous procedure, 39 of them can be estimated by SAM-6D model, without exceeding the video memory limit. 31 of them passed the 5 degrees and 5 centimeters standard. Each data instance serves as a comprehensive representation of the scene, containing multimodal sensory information critical for 6D pose estimation. The dataset composition is detailed in Table 3.2:

Name	Explain
depth_image	A two-dimensional matrix of depth images, where each pixel value represents the distance from an object in the corresponding scene to the camera.
depth_intrinsics	Describe the internal parameter matrix of the depth camera as a 3x * 3 matrix.
color_image	Standard RGB three channel image, providing color and texture information of the scene.
color_intrinsics	Describe the internal parameter matrix of a color camera as a 3 * 3 matrix.
rotation_matrix	A 3 * 3 matrix representing the rotational posture of an object or camera in three-dimensional space

Table 3.2: Data information

The depth-color data pairing enables multimodal fusion approaches in pose estimation. Depth intrinsics and color intrinsics are stored separately to accommodate potential sensor calibration differences. Rotation matrices are recorded in both object-centric and camera-centric coordinates to support different algorithm requirements. This comprehensive data structure ensures compatibility with various 6D pose estimation frameworks while maintaining flexibility for different processing pipelines.

3.3 Experiment and Optimization

3.3.1 YOLO-6D

The experimental process of this study mainly includes three key stages: data preparation, model training, and performance optimization. The iterative optimization experimental strategy is adopted to gradually improve the model performance. This approach aligns with common practices in deep learning research where gradual refinement often yields better results than single-step optimization.

3.3.1.1 Data generation and preprocessing

Build a synthetic dataset based on BlenderProc2 physics rendering engine, which has become a standard tool in computer vision for generating photorealistic training

data. Generate 2000 sets of multimodal training samples through parameterized scene configuration. Each sample contains:

- RGB image (PNG format, 24 bit color depth): Captures surface texture and color information essential for visual recognition
- Depth map (16 bit floating-point storage): Provides geometric information for spatial understanding
- 6D pose annotation file (JSON format, including rotation matrix and translation vector): Contains precise spatial coordinates required for pose estimation
- Instance segmentation mask (generated by boptoolkit with a resolution of 640 × 480): Enables accurate object boundary detection

The dataset was divided using stratified sampling, with a 3:1 ratio of 1500 training sets and 500 testing sets. This division ensures balanced data distribution between training and validation phases. To enhance data generalization, there are random perturbations in the lighting effects in the dataset, simulating real-world environmental variations that improve model robustness.

3.3.1.2 Model training configuration

The experimental platform adopts NVIDIA RTX4070 GPU (12GB GDDR6X graphics memory), which provides sufficient computational power for medium-scale deep learning tasks. Implement model training based on PyTorch 1.12 framework, leveraging its automatic differentiation and GPU acceleration capabilities. The main parameter configurations are as follows:

- Input resolution: 640 × 480 (maintaining BOP benchmark dataset specifications to ensure compatibility)
- Training cycle: 100 epochs (balance between training efficiency and model convergence)
- Optimizer: Momentum SGD (momentum=0.9, initial learning rate 3e-4): Combines gradient descent momentum for stable convergence
- Composite loss (including classification cross entropy, IoU loss, and pose regression L1 loss): Addresses multiple learning objectives simultaneously
- Batch processing scale: 8 (limited by video memory capacity): Maximizes GPU utilization under hardware constraints

During the training process, cosine annealing learning rate scheduling is used to dynamically adjust learning rates, and validation set accuracy evaluation is performed every 10 epochs to monitor overfitting.

ONNX Runtime is a high-performance inference engine used to run machine learning models based on ONNX (Open Neural Network Exchange) format. This framework plays a crucial role in deployment phase by enabling cross-platform compatibility. It is developed by Microsoft and open-source, with the goal of efficiently executing machine learning models on multiple platforms and hardware. This tool can run on multiple platforms and supports multi-hardware acceleration. ONNX models can come from various frameworks such as PyTorch, TensorFlow, XGBoost, etc., and have good compatibility. After model convergence, format conversion is performed through ONNX Runtime to achieve cross-platform deployment, which simplifies the transition from research to production environments.

3.3.1.3 Performance optimization

Preliminary experiments have shown that the model exhibits significant bias in attitude estimation tasks. Through multidimensional attribution analysis, implement the following optimization measures:

1. Appearance consistency enhancement: Addressing the significant differences between the appearance parameters of the reconstructed target object and the actual part image. Reconfigure the physics based rendering (PBR) process in BlenderProc2 to ensure that specular and diffuse properties match the real environment, thereby improving the realism of materials.
2. Scene interference suppression: Adopting a background replacement strategy to replace the original random scene with a single material that is closer to the color of the console. This reduces background noise interference. Meanwhile, simplify the dataset images to only contain a single component, minimizing unnecessary visual complexity.
3. Training strategy optimization: Extend the training cycle to 150 epochs to allow fuller parameter convergence. Add gradient clipping (max_norm=1.0) to prevent gradient explosion, enhancing training stability.

After secondary verification, it was found that the model improvement was limited. The following are two possible key limiting factors:

1. Scale sensitivity issue: The difference in aspect ratio between the test image resolution (1280×720) and the training size (640×480) leads to geometric distortion, affecting feature alignment.
2. Model capacity limitation: YOLO-6D lacks fine-grained feature extraction for complex mechanical parts. Its architecture cannot recognize the geometric features of the parts to be assembled, limiting precision in industrial scenarios.

Based on the above analysis, this study ultimately adjusted the technical route and developed the SAM-6D model with better steering accuracy. This transition demonstrates the importance of flexible strategy adjustment when facing technical bottlenecks in practical engineering applications.

3.3.2 SAM-6D

This study adopts SAM-6D as the core framework for 6D attitude estimation. The high precision and strong adaptability of this method are particularly valuable in industrial scenarios, where obtaining labeled training data for specialized components is challenging. This ability enables the framework to maintain excellent generalization ability even without target object training data. In the specific implementation process, we adopted a layered computing optimization strategy to solve the memory bottleneck problem caused by large-scale point cloud matching. This optimization is crucial because traditional pose estimation methods often cannot effectively scale with increasing point cloud density.

We also discovered a key technical challenge when using SAM-6D as the core framework for 6D pose estimation. Many of the original methods rely on GPU for complete matrix calculations, and in bit pose estimation models, there is often a shortage of GPU memory. This limitation stems from the fundamental computational characteristics of point cloud feature matching. The template point cloud N and scene

point cloud M are both very large-scale, resulting in the constructed pairwise distance matrix D reaching tens or even millions of dimensions. To quantify this challenge, consider a typical scenario: when $M = 2048$ and $N = 2048$, with each point represented as a three-dimensional float32 vector, the memory required for a single calculation is:

$$\text{Memory}(D) = M \times N \times \text{dtype\_size} = 2048 \times 2048 \times 4\,\text{Bytes} = 16.78\,\text{MB} * batch \tag{3.2}$$

This calculation reveals the quadratic growth pattern of memory consumption relative to point cloud size. However, in practical calculations, the demand becomes more severe due to the need to process multiple sets of geometric transformation candidates simultaneously. When the number of candidate poses reaches 1024, the memory requirement escalates to 17.18 GB, far exceeding the graphics memory capacity of conventional GPUs (such as the 12 GB of NVIDIA RTX 4070). This bottleneck fundamentally limits the scalability of traditional GPU-based implementations.

To address this critical challenge, we designed a heterogeneous computing allocation strategy that intelligently distributes computational tasks between GPU and CPU. The specific implementation adopts a block computing system with three-stage processing:

1. Feature Extraction: Parallel extraction of point cloud features is completed on the GPU side, leveraging its massive parallel computing advantage for initial data processing.
2. Block Computation: The feature matrix blocks are transmitted to the CPU for distributed distance calculation, utilizing CPU memory resources to handle large-scale matrix operations.
3. Result Integration: The calculation results are transmitted back to the GPU through memory mapping for final pose optimization, combining the precision of GPU computing with the capacity of CPU memory.

This hybrid method effectively balances computational efficiency and memory limitations. By strategically allocating tasks based on hardware strength, we can maintain computational throughput while avoiding memory overflow. The effectiveness of this strategy was verified through the system experiments discussed in Section 4.3. Compared with traditional implementations, we found that while improving memory efficiency, the accuracy of attitude estimation decreased. Nevertheless, this optimization still provides ideas for implementing large-scale point cloud processing algorithms on resource constrained hardware platforms.

3.4 6D pose estimation process

In this system, we attempt to use ROS2 (Robot Operating System 2) as the communication and task scheduling framework, combined with the SAM-6D model, to achieve high-precision 6D pose recognition of the target object. ROS2 has been upgraded based on its predecessor ROS, enhancing real-time performance and security features that are crucial for industrial applications. As an open-source framework

and toolkit designed for robot development, ROS2 aims to simplify the construction process of complex robot systems through standardized interfaces and modular architecture. Although the name includes' operating system ', it serves as a middleware platform running on existing operating systems such as Ubuntu, providing a platform for hardware interaction and inter process communication. This design concept achieves seamless integration of heterogeneous components in robot systems. ROS2 provides a modular and distributed node communication mechanism, enabling efficient and reliable collaboration between image acquisition, inference processing, and control execution. The adoption of Data Distribution Service (DDS) as the underlying communication protocol ensures deterministic data transmission, which is crucial for time-sensitive robotic operations. The main process of the system is shown in Figure 3.5, illustrating the data flow from perception to action execution.

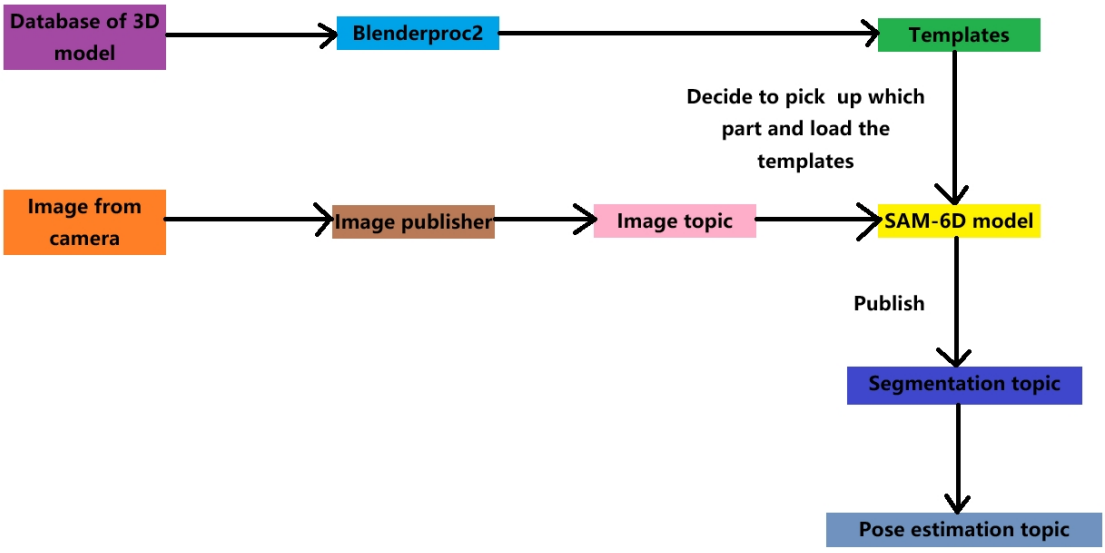


Figure 3.3: Estimation process

The system is based on the ROS2 platform to achieve 6D pose recognition. Firstly, BlenderProc2 is used to generate image templates of the target object from the 3D model database; The images captured by the camera are sent to the image topic through the image publishing node. The SAM-6D model node subscribes to the images and loads the corresponding templates to perform target segmentation and pose estimation; The final result is published through ROS2's segmentation of topics and pose estimation topics for downstream tasks.

3.4.1 3D model generation recognition template

The template generation stage forms the foundation for subsequent recognition tasks. When our database already stores 3D models of various parts (such as. obj or. stl files), we utilize BlenderProc2's batch processing capabilities to systematically generate template images. Due to its accurate simulation of materials and lighting in physics, this open-source rendering tool is particularly suitable for industrial applications. For each component, the generation process will generate:

- RGB images with varying illumination conditions
- Depth maps aligned with robotic sensor specifications
- Multi-view projections covering typical observation angles

These template files are organized in a structured directory within the local database, ensuring efficient retrieval during real-time operations. Each part corresponds to an independent template set, allowing the system to handle multiple component types simultaneously without interference.

3.4.2 Instruction sending and template loading

When the robotic arm receives the task of grasping a specific part, the system initiates a coordinated workflow through ROS2's service-client architecture. Control instructions are transmitted to the SAM-6D model node via dedicated ROS2 topics, specifying the required component type and corresponding template folder. This instruction protocol contains data such as:

- Component identification code
- Priority level for time-critical tasks
- Geometric constraints for grasping operations

In ROS2, this process is implemented through nodes and topics. The decoupled architecture allows independent development and testing of each functional module. The template loading mechanism employs ROS2 parameters to dynamically configure the recognition pipeline, enabling rapid switching between different component templates without system restart. This flexibility is essential for adaptive manufacturing scenarios requiring frequent product changeovers.

3.4.3 Image acquisition and ROS2 topic release

The visual perception subsystem plays a crucial role in maintaining situational awareness. The RGB-D camera installed on the end effector of the robot arm captures synchronized RGB and depth data at a certain frequency, ensuring temporal alignment between color and geometric information. Through the optimization of the image transport layer in ROS2, the image _publisher node performs the following operations:

- Sensor data timestamp synchronization
- Automatic white balance calibration
- Depth map noise filtering

Processed data streams are published to standardized topics:

- /Image_raw: RGB images encoded in sensor_msgs/Image format
- /Depth_image: Depth information stored as sensor_msgs/PointCloud2

These topics implement Quality of Service policies to prioritize real-time data delivery. The SAM-6D model nodes maintain persistent subscriptions, utilizing ROS2's callback mechanism to trigger processing upon new data arrival.

3.4.4 SAM-6D node processing

The SAM-6D node embodies the core intelligence of the system, integrating computer vision and spatial reasoning capabilities. Upon receiving image data, the

processing pipeline executes:

1. **Image Segmentation**: Segmented Arbitrary Model (SAM) utilizes visual features and spatial relationships to segment part pixels from RGB images.
2. **Template Matching**: The SAM-6D model exports templates from a preloaded CAD database and matches them with the segmented regions.
3. **Pose Estimation**: The SAM-6D model uses a two-stage estimator to first calculate the rough direction of the segmented features, and then refine the calculation results.

This hybrid approach balances speed and precision, achieving millimetre-level accuracy required for mechanical assembly.

3.4.5 ROS2 multi topic output

Post-processing results are disseminated through ROS2's pub-sub mechanism to various downstream consumers:

- **pose_estimation**: 6D pose (position + rotation) in robot base frame
- **confidence_score**: Matching certainty level (0-1 scale)
- **pointcloud_segmented**: Filtered point cloud of target object

Motion planning nodes utilize these outputs to:

- Compute collision-free trajectories
- Select optimal grasping points
- Adjust tool orientation for insertion tasks

The distributed architecture ensures that computational intensive tasks (e.g., point cloud processing) remain isolated from real-time control loops, enhancing system stability and responsiveness.

4

Results

4.1 YOLO-6D estimation result

In our experiment, we initially used YOLOX as the baseline framework for object detection tasks. Through multiple rounds of parameter adjustment and optimization attempts, including system adjustment of input image size, we observed that YOLO-6D still performed poorly on the collected real dataset. Even after optimizing the parameters mentioned earlier, this performance flaw still exists, as shown in the visualized recognition results in Figure 4.1. As shown in the figure, the YOLO-6D model experienced significant errors during instance segmentation, and the 6D pose estimation results were also inaccurate.

Figure 4.1: YOLO-6D recognition result

The complete training process encompassed 100 full epochs, with detailed loss metrics recorded at each optimization step. Specifically, the progression of six critical loss components was meticulously documented: the comprehensive total loss shown in Figure 4.2, the category-specific classification loss in Figure 4.3, the prediction reliability-focused confidence loss in Figure 4.4, the spatial coordinate regression loss in Figure 4.5, the segmentation-related mask loss in Figure 4.6, and the geometrically constrained edge loss in Figure 4.7. Each loss component exhibits distinct convergence characteristics throughout the training phases.

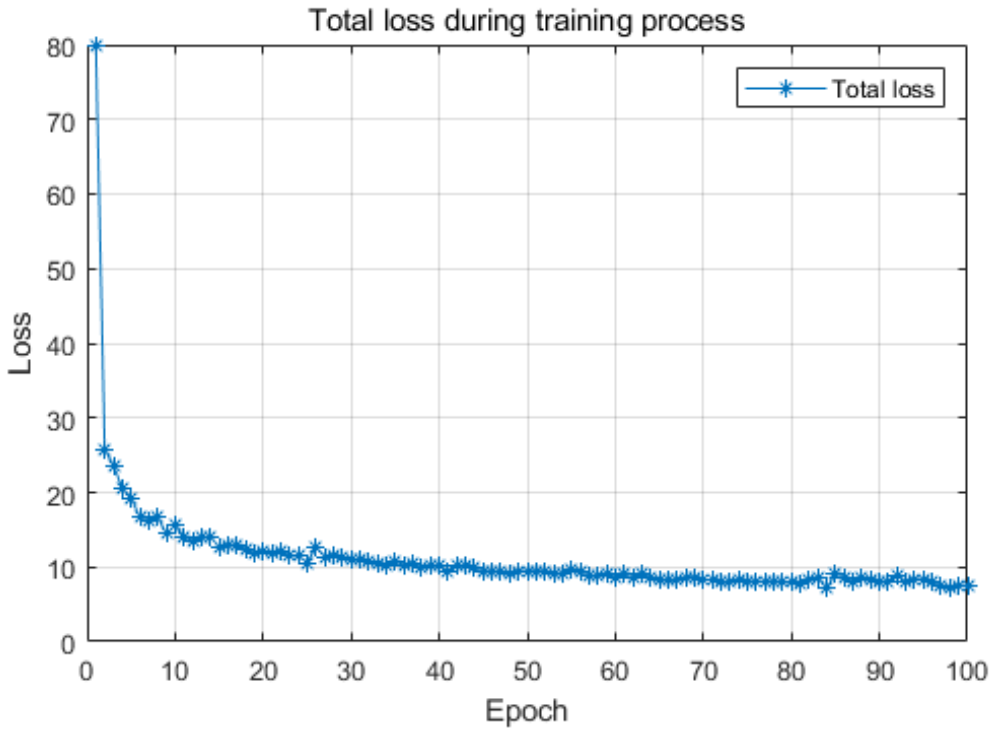


Figure 4.2: Total Loss function

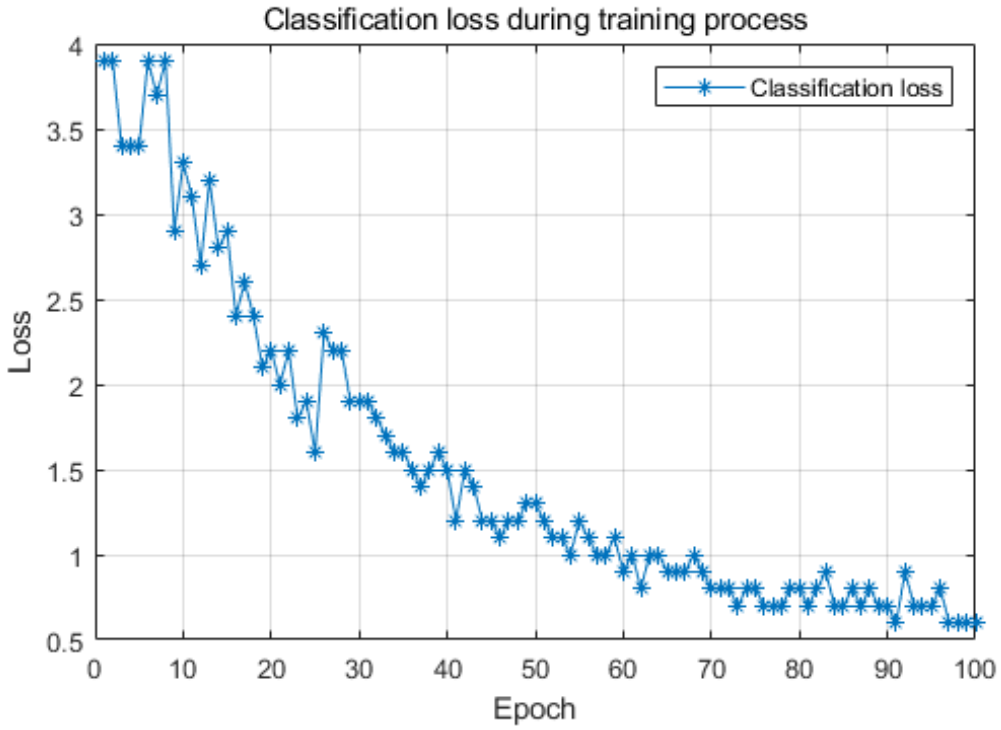


Figure 4.3: Classification loss function

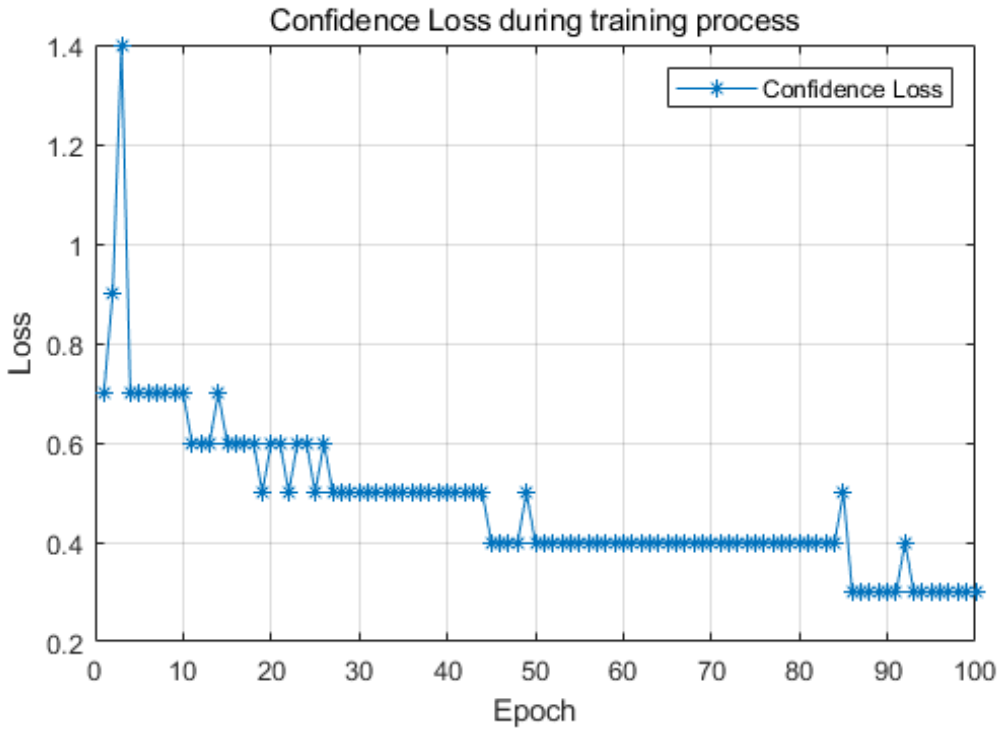


Figure 4.4: Confidence Loss function

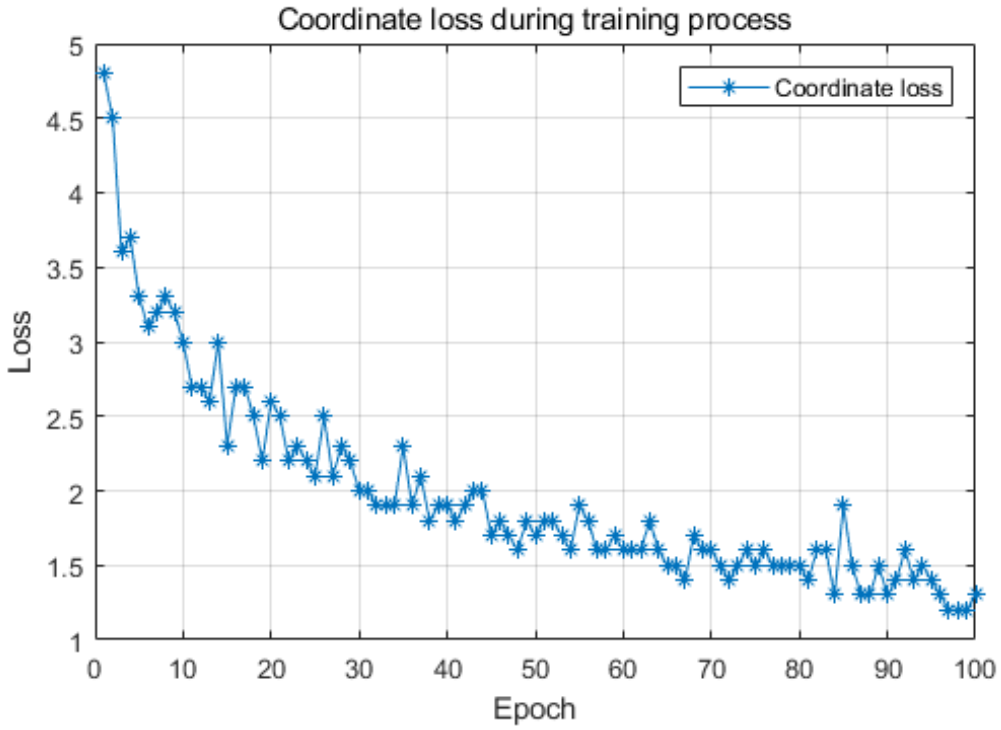


Figure 4.5: Coordinate loss function

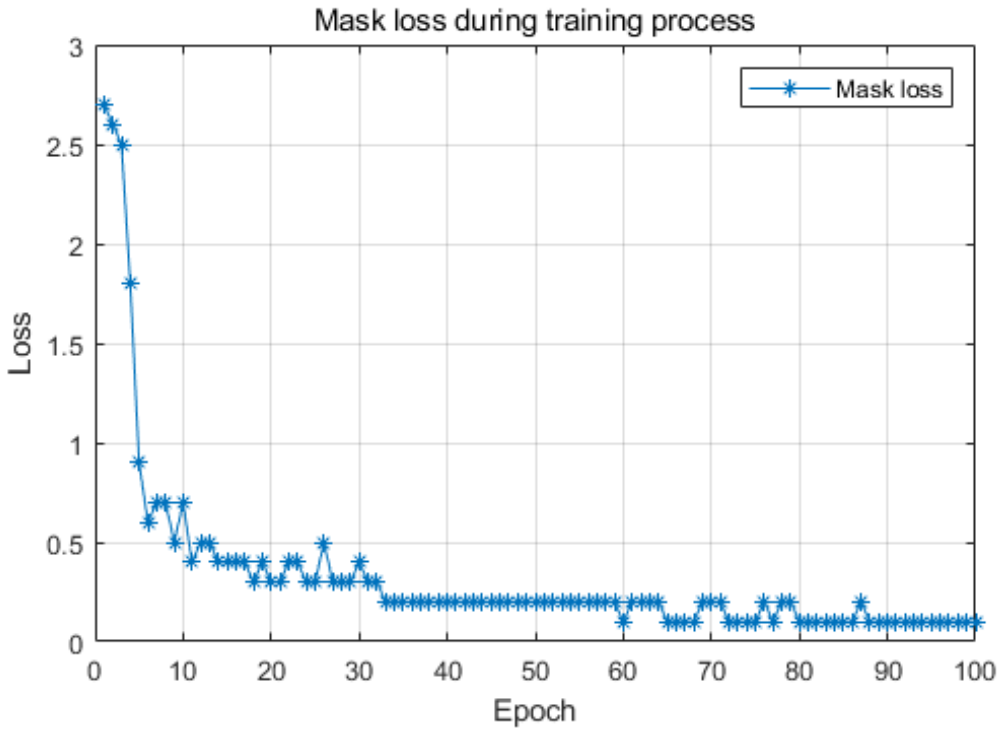


Figure 4.6: Mask loss function

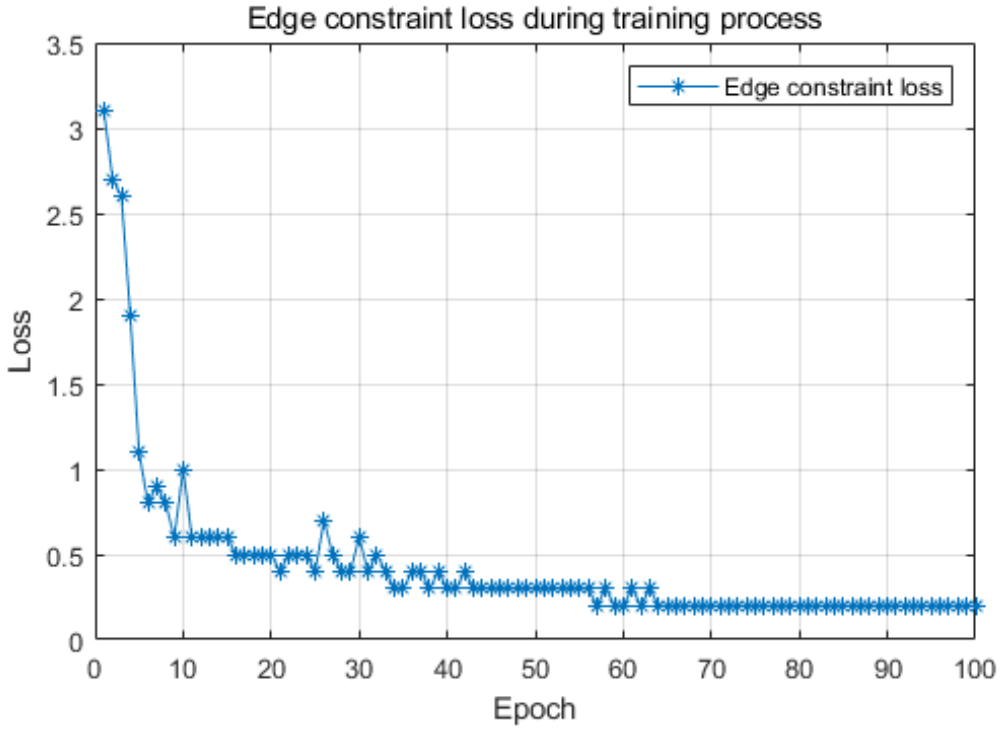


Figure 4.7: Edge constraint loss function

Although all loss curves roughly show the expected downward trajectory during the model optimization process, this numerical improvement has not been translated

into satisfactory actual performance. It can be observed that the total loss, mask loss, and edge constraint loss show a decreasing trend during the training process. However, although the classification loss, confidence loss, and coordinate loss also decrease during the training process, they all experience severe oscillations, especially the confidence loss. The results indicate that during the training process, as the gradient decreases, it cannot be guaranteed that all loss functions will gradually decrease, which also leads to inaccurate pixel segmentation of YOLO-6D. Three main factors have been identified as potential reasons for poor model performance. Firstly, the complex geometric shape of the target component (the model has many edges, corners, and faces. The color of the model is confused with the surrounding markings) poses significant recognition challenges, especially in accurate 3D bounding box estimation. Secondly, in the preprocessing stage, necessary image resizing operations inadvertently caused shape distortion, resulting in a mismatch between the training data features and the actual input patterns. For example, if the size of the input image is 1280 * 720 pixels, during the input process of the model, the image will be stretched to 640 * 480 pixels. Due to the different ratio of length to width, the shape of an object can become distorted. Thirdly, the inherent domain gap between the synthetic training dataset and the real-world deployment environment fundamentally limits the model's generalization ability.

Notably, while each loss component achieved stable convergence individually, their combined effect failed to produce the desired recognition accuracy. This suggests potential limitations in the loss weighting strategy or deeper architectural constraints. The persistent performance gap highlights the need for more sophisticated approaches in handling complex geometric structures and domain adaptation challenges.

4.2 SAM-6D estimation result

Following a comprehensive review of relevant literature, we identified the SAM-6D methodology and selected it for implementation. Empirical observations demonstrate that this approach achieves superior performance in object segmentation tasks, exhibiting enhanced capability in precisely delineating object boundaries within image datasets compared to our prior methodologies. The segmentation contours generated by SAM-6D in virtual environments are presented in Figure 4.8, with corresponding point cloud reconstructions and three-dimensional bounding box estimations shown in Figure 4.9. Furthermore, Figure 4.10 illustrates the model's segmentation performance in real-world scenarios, accompanied by its associated point cloud visualization and 3D bounding box determination in Figure 4.11. Whether it is contour segmentation, point cloud modeling, or 3D bounding box calculation, good results have been achieved in both simulated and actual datasets.

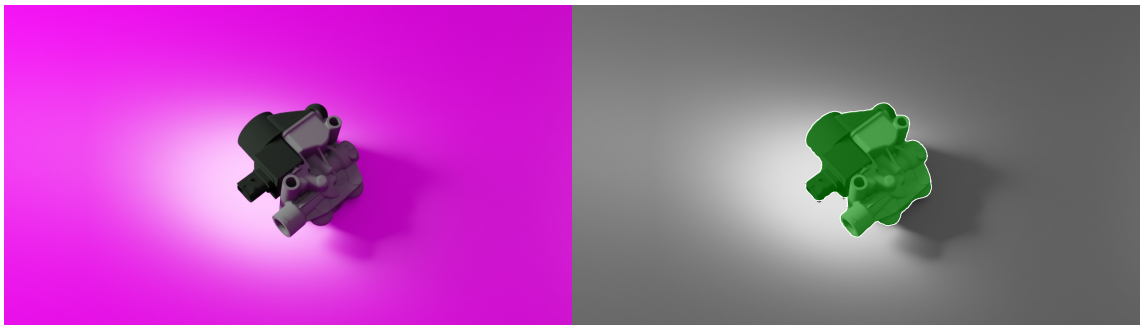


Figure 4.8: Outline in virtual dataset

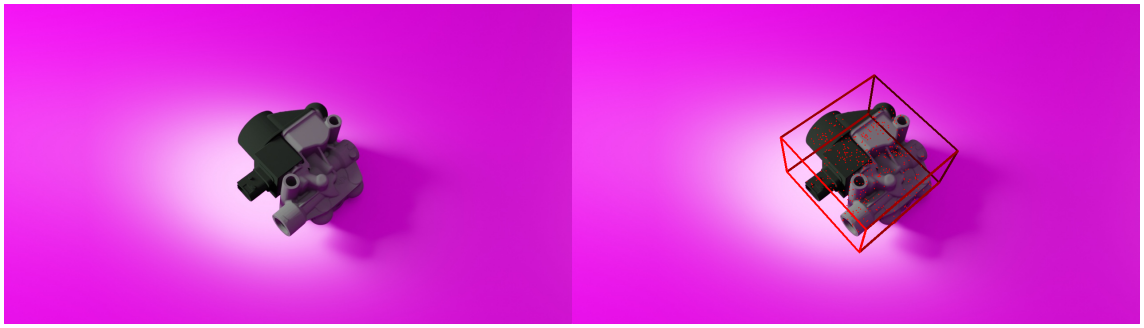


Figure 4.9: Point cloud and 3D bounding box in virtual dataset

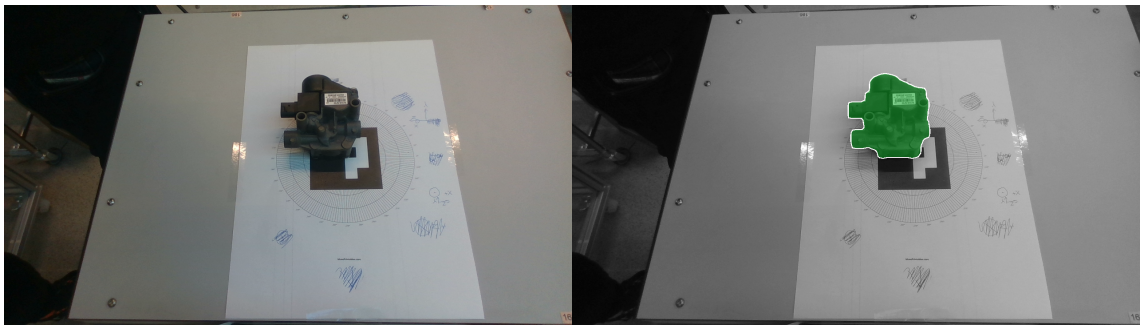


Figure 4.10: Outline in actual environment

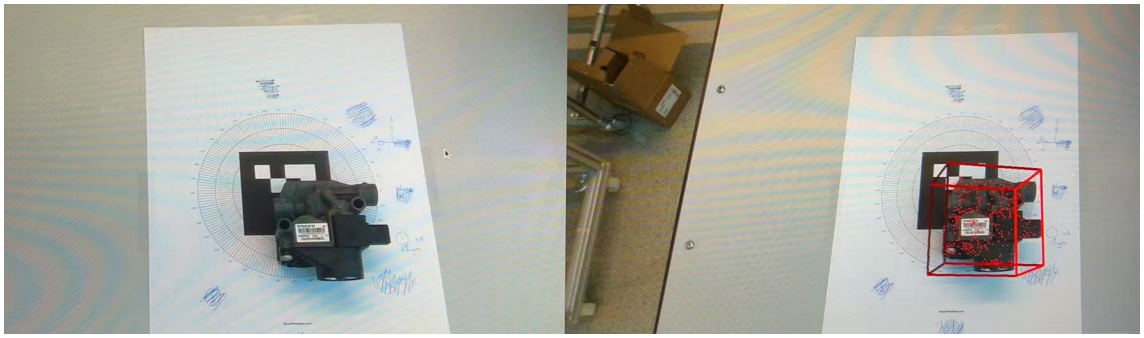


Figure 4.11: Point cloud and 3D bounding box in actual environment

In order to systematically evaluate the 6D attitude estimation capability of the SAM-6D algorithm, a rigorous experimental framework was established using a synthetic dataset consisting of 40 industrial component models. This method adopts a dual-mode input architecture to synchronously process RGB images (640 × 480 resolution) and registered depth maps. According to the standardized 6D pose assessment protocol, a simulated environment was constructed to combine variable lighting conditions and diverse background environments. Each experimental instance was annotated with precise ground truth attitude data.
The evaluation metrics were defined through a dual-parameter criterion: rotational discrepancy (θ) and translational deviation (t). The rotational error was derived from the minimum angular difference between predicted and ground truth rotation matrices via Lie algebra mapping, formulated as:

$$\theta = \arccos\left(\frac{\mathrm{Tr}(\mathbf{R}_{\mathrm{pred}}\mathbf{R}_{\mathrm{gt}}^{\top}) - 1}{2}\right) \tag{4.1}$$

Where $\mathbf{R}_{\mathrm{pred}}$ indicates the predicted rotation matrix and $\mathbf{R}_{\mathrm{gt}}$ indicates the real rotation matrix. $Tr()$ caculates the trace of the matrix.
Translational accuracy was quantified through the L2-norm distance between predicted and ground truth displacement vectors:

$$t = \|\mathbf{t}_{\mathrm{pred}} - \mathbf{t}_{\mathrm{gt}}\|_2 \tag{4.2}$$

Where $\mathbf{t}_{\mathrm{pred}}$ indicates the predicted transition and $\mathbf{t}_{\mathrm{gt}}$ indicates real transition.
A successful estimation was defined as meeting the dual thresholds of $\theta \leq 5^{\circ}$ and $t \leq 5cm$. Quantitative results of the evaluation are presented in Table 4.1.

ID	Rotation Error (°)	Translation Error (mm)	Success (5cm/5°)
1	3.012	5.131	Yes
2	0.256	1.036	Yes
3	0.571	1.715	Yes
4	0.486	1.838	Yes
5	0.274	1.482	Yes
6	0.21	0.992	Yes
7	0.334	1.523	Yes
8	0.603	0.736	Yes
9	0.162	1.353	Yes
10	0.392	0.933	Yes
11	0.459	1.11	Yes
12	1.11	2.038	Yes
13	0.661	2.334	Yes
14	0.156	1.119	Yes
15	0.269	0.7	Yes
16	0.844	1.672	Yes
17	124.814	117.424	No
18	0.456	1.199	Yes
19	0.058	1.163	Yes
20	0.414	1.202	Yes
21	0.373	1.268	Yes
22	0.505	1.22	Yes
23	0.078	1.251	Yes
24	0.36	1.558	Yes
25	0.612	1.593	Yes
26	0.972	2.583	Yes
27	0.288	1.932	Yes
28	0.187	1.547	Yes
29	0.49	1.139	Yes
30	0.446	1.149	Yes
31	0.646	1.258	Yes
32	0.709	1.841	Yes
33	0.755	1.254	Yes
34	0.276	1.814	Yes
35	0.68	1.244	Yes
36	0.655	1.929	Yes
37	0.163	1.35	Yes
38	0.119	1.197	Yes

Table 4.1: Pose evaluation results in simple environment

The The rotation error and translation error are shown in Figures 4.12 and 4.13, respectively.

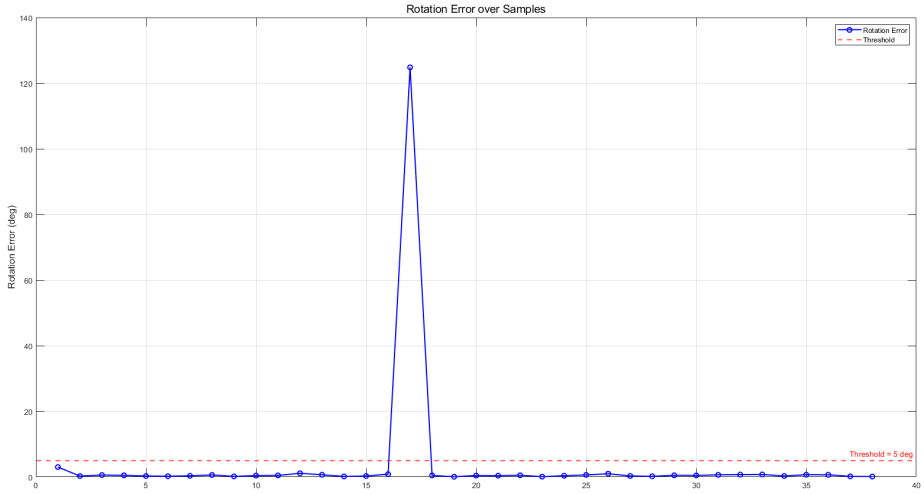


Figure 4.12: Rotation error in simple environment

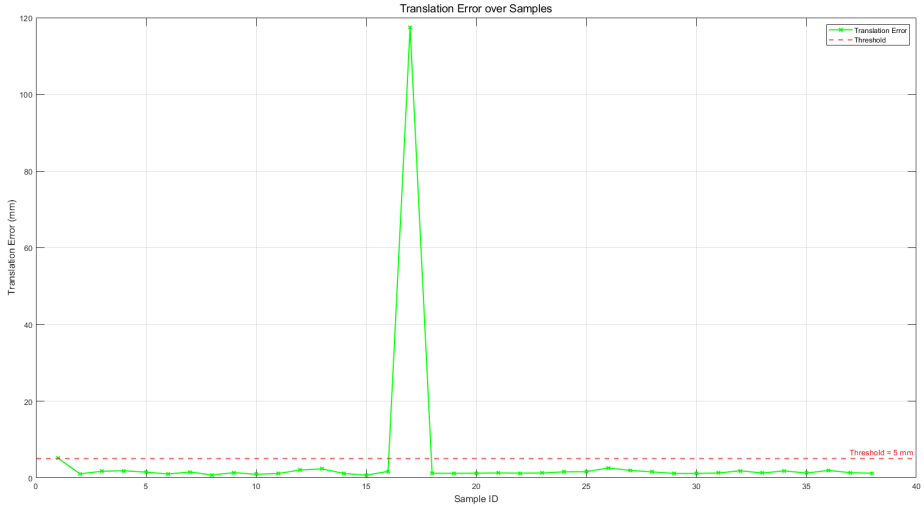


Figure 4.13: Translation error in simple environment

Within the experimental dataset of 40 images, two instances were excluded due to GPU memory constraints exceeding hardware limitations. Among the remaining 38 valid samples, 37 demonstrated simultaneous compliance with both evaluation criteria. Comparative analysis reveals that the SAM-6D model exhibits significantly higher pose estimation accuracy than the YOLO-6D baseline, demonstrating 97% joint compliance rate in valid samples and enhancing its applicability in industrial settings. Quantitative error analysis indicates a mean rotational error of 0.515° (variance: 0.240°) and mean translational error of 1.424 cm (variance: 0.175 cm)

across compliant cases. Error attribution studies suggest these deviations primarily originate from extrinsic parameter discrepancies in multi-sensor configurations. While the simulation environment assumes identical intrinsic parameters for RGB and depth sensors, extrinsic misalignment (particularly in rotational and translational offsets) may induce feature registration artifacts.

It is noteworthy that two additional failure cases involving high-complexity assemblies were excluded from statistical analysis due to memory overflow constraints, highlighting the necessity for memory optimization in future implementations. Furthermore, the performance gap between synthetic data and real-world sensing conditions (e.g., ToF camera depth quantization errors, multipath interference) requires systematic investigation, as these factors may substantially impact algorithmic robustness during physical deployment — a critical focus area for subsequent research.

4.2.1 Accuracy of SAM-6D model in Complex Environments

To replicate real-world industrial operating conditions, three interference objects were systematically selected from the T-LESS benchmark dataset and incorporated into the experimental setup. Concurrently, textured backgrounds resembling metallic and wooden surfaces were intentionally selected to rigorously evaluate the algorithm's robustness against environmental variability. The contour segmentation performance of the SAM-6D model under these complex interference conditions is presented in Figure 4.14, while the reconstructed point cloud topology and corresponding 3D bounding box projections are depicted in Figure 4.15.

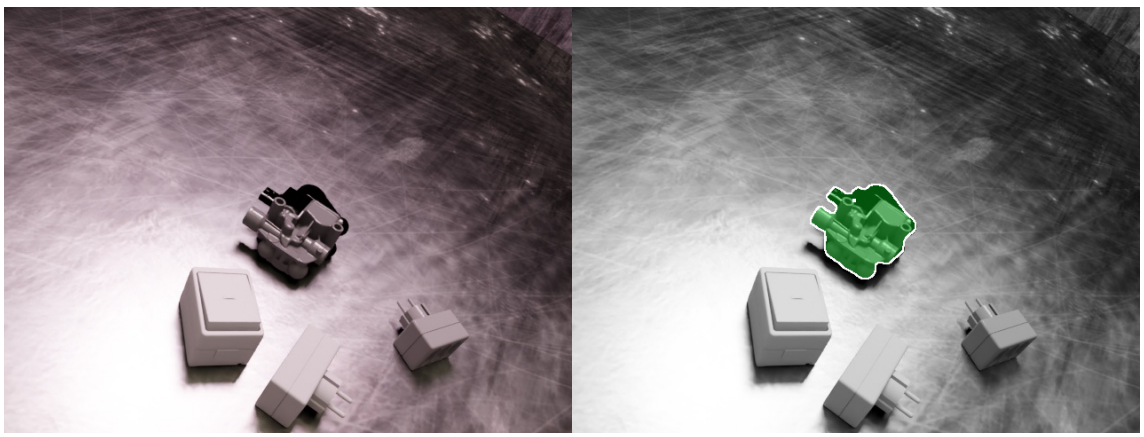


Figure 4.14: Outline in complex environment

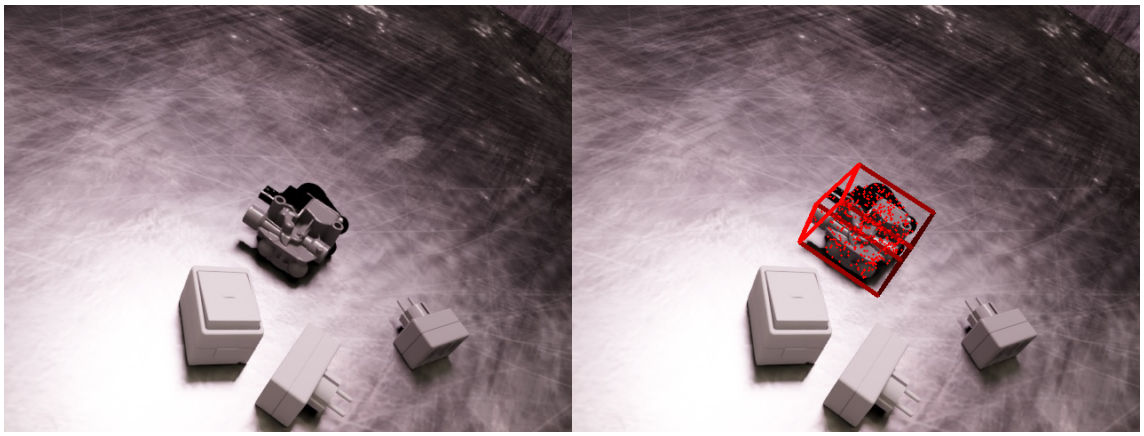


Figure 4.15: Point cloud and 3D bounding box in complex environment

A test cohort comprising 40 data samples was established for performance validation, of which 6 samples (15%) were excluded due to GPU memory overflow constraints. This exclusion demonstrates a positive correlation between environmental complexity and computational resource demands. Among the remaining 34 valid samples, three samples (8.8%) failed to satisfy the dual error thresholds (rotational error less than 5 degree and translational error less than 5 centimeter), yielding a model accuracy rate of 91% under operational conditions. Detailed error analysis revealed the rotational error distribution exhibited a mean of $0.756°(\sigma^2 = 0.364°)$, while translational errors averaged 1.677 cm ($\sigma^2 = 0.295cm$) across compliant cases. The complete experimental measurements are tabulated in Table 4.2.

ID	Rotation Error (°)	Translation Error (mm)	Success (5cm/5°)
1	143.495	199.750	No
2	0.522	0.494	Yes
3	0.502	0.820	Yes
4	0.183	0.428	Yes
5	0.408	0.496	Yes
6	0.807	0.641	Yes
7	0.450	1.187	Yes
8	0.423	0.697	Yes
9	0.460	0.710	Yes
10	0.265	0.819	Yes
11	156.824	38.231	No
12	0.098	0.747	Yes
13	0.230	0.270	Yes
14	0.358	0.470	Yes
15	0.387	0.829	Yes
16	0.109	0.407	Yes
17	0.838	0.749	Yes
18	0.490	0.552	Yes
19	0.412	0.758	Yes
20	1.343	1.736	Yes
21	0.492	0.480	Yes
22	0.701	0.206	Yes
23	110.924	134.650	No
24	0.217	0.849	Yes
25	0.490	0.420	Yes
26	0.163	0.605	Yes
27	0.664	0.963	Yes
28	0.152	0.618	Yes
29	0.528	0.852	Yes
30	0.466	0.844	Yes
31	0.491	0.873	Yes
32	0.478	0.261	Yes
33	0.299	0.262	Yes
34	0.713	0.931	Yes

Table 4.2: Pose evaluation results in complex environment

The rotation error is shown in Figure 4.16 and translation error is shown in Figure 4.17.

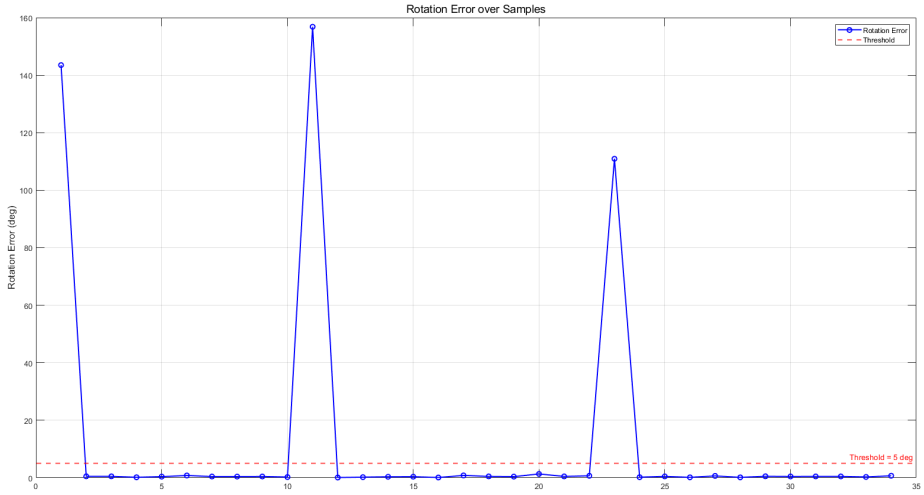


Figure 4.16: Rotation error in complex environment

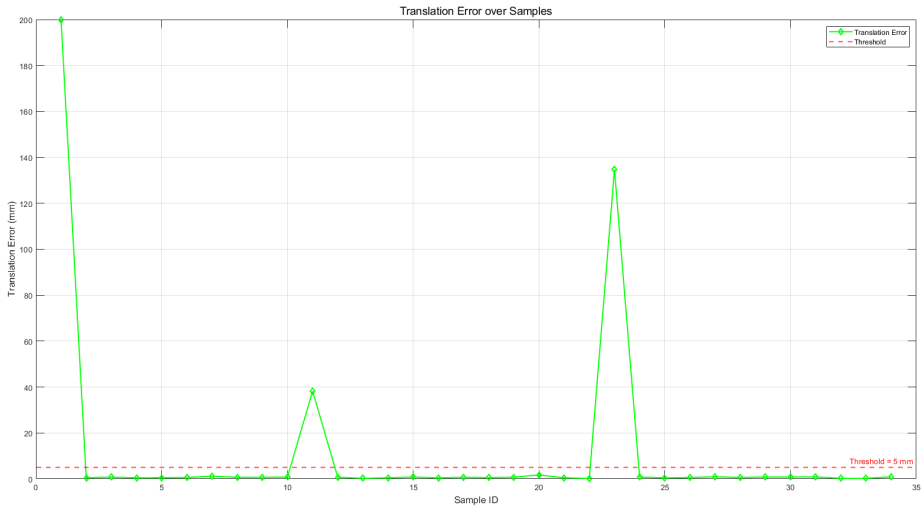


Figure 4.17: Translation error in simple environment

Notably, the synthetic dataset was specifically designed to incorporate intentional part-interference occlusion scenarios, effectively emulating workpiece obscuration patterns observed in industrial production environments. Figure 4.18 illustrates the model's contour segmentation capability under partial occlusion conditions, while Figure 4.19 demonstrates the corresponding 3D point cloud reconstruction and bounding box determination in such challenging scenarios. Quantitative evaluation of these occlusion cases revealed rotational error measurements of 1.343° and translational deviations of 1.736 cm — both maintaining compliance with the dual

error thresholds. This empirical evidence validates the operational viability of the SAM-6D framework for 6D pose estimation in factory automation contexts where visual occlusion constitutes a critical operational constraint.

Figure 4.18: Outline in overlapped data

Figure 4.19: Point cloud and 3D bounding box in overlapped data

4.3 Video Memory Optimization Strategy

To address the issue of high graphics card memory requirements, we reallocated the model's computational resources. This strategy reduces peak video memory usage; however, experimental results show that excessive computational offloading significantly compromises the accuracy of pose estimation (see the comparison results in Figure 4.20 and Figure 4.21). When some computing tasks are assigned to the CPU, there are significant errors in the calculation of the 3D bounding box of the object.

Figure 4.20: Initial recognition result

Figure 4.21: Improve recognition results

4.3.1 Calculation accuracy analysis

Comparative experiments revealed a significant decline in pose estimation accuracy when the proportion of CPU-based computation was high. A multidimensional attribution analysis identified the following primary factors contributing to this accuracy loss:

1. Cumulative error of numerical accuracy: The float32 operation of GPU is implemented using SIMD architecture, and its hardware acceleration characteristics of transcendental functions (such as sqrt and atan2) make the error accumulation rate lower than that of CPU.
2. Sensitivity to data chunk order: The lack of asynchronous execution guarantee from CUDA Stream in CPU side chunk operations results in index offset during chunk reassembly, which in turn affects the selection of topk candidate poses.
3. Parallel optimization difference: The GPU side fused kernel integrates matrix operations, normalization, and other operations into a single atomic operation, while the CPU side's layer by layer computation chain increases the RMS error of intermediate results.
4. Non deterministic computation differences: torch.mm adopts the deterministic mode of MKL on the CPU side, while there are non deterministic fluctuations when TF32 tensor kernel is enabled on the GPU side, resulting in biased attitude voting results.

By employing dynamic load balancing algorithms, computation during the feature matching stage is retained on the GPU, while CPU-GPU collaborative processing is applied only in the post-processing stage. This approach enhances computational

efficiency but results in a reduction in model recognition accuracy. For industrial deployment, it is recommended to use computing devices with a minimum of 24GB of video memory to ensure that all graphical computations can be fully executed on the GPU.

4.4 Future work

4.4.1 ROS based integration framework

The SAM-6D model is capable of providing accurate 6D pose estimation (3D translation and 3D rotation) for industrial components, serving as a key input for the robotic arm grasping process. For the final project, a modular architecture will be designed on the ROS platform, primarily consisting of three nodes.

- SAM-6D node: Publish the estimated 6D pose and aligned point cloud of the target part.
- Capture planning nodes: subscribe to pose and point cloud data, combine geometric analysis (such as mapping capture points) and 3D bounding boxes of parts to generate collision free capture poses.
- Motion Control Node: Utilizing the robotic arm's kinematic and dynamic models, the grasping pose is converted into joint trajectories.

These three nodes have been already finished by us and another group individually. However, the data transmission between the three nodes still needs further debugging. Meanwhile, the issue of errors that arise during data transmission needs to be further addressed.

4.4.2 Key outputs and their role in grasping

6D attitude information This encompasses translation, which directly defines the part's position within the robotic arm's base coordinate system through coordinate transformation, and rotation, which determines the part's orientation to align the gripper with its approach axis—such as vertical gripping with suction cups or horizontal gripping with parallel grippers.

Align point cloud model Provide dense geometric details on the surface of the parts, supporting the following functions. The first one is selecting stable grasping points through analysis of local curvature and contact area.The second one is planing collision free paths based on the occupancy map of the robotic arm.

3D bounding box As a spatial prior for rough positioning, it accelerates grasping planning in cluttered scenes, and combines 6D pose to define the occupied volume of parts in the workspace to assist motion planning.

4.4.3 Challenges and Importance

4.4.3.1 Develop End-to-End Framework to Reduce Error Propagation

The current SAM-6D workflow comprises sequential modules (segmentation, semantic matching, geometric verification, pose regression). While each module can be

independently optimized, intermediate uncertainties may cause cumulative errors. Future research should integrate SAM's mask output with pose regression into an end-to-end network, enabling joint optimization in a unified feature space to enhance stability and inference speed.

4.4.3.2 Build Lightweight Models for Edge Deployment

Existing SAM and DINOv2 architectures demand high computational complexity, GPU memory, and processing power, hindering deployment on embedded platforms (e.g., mobile robots, AGVs, low-power industrial cameras). Future work should adopt lightweight alternatives (e.g., FastSAM, MobileSAM) or employ model pruning, distillation, and quantization to create compact industrial vision models.

4.4.3.3 Implement Adaptive Keypoint Selection and Geometric Constraints

The fixed point-cloud IoU matching strategy struggles with occlusions and complex geometries. Solutions include point-region attention mechanisms via Transformer modules or learnable geometric validators to improve CAD model alignment. Self-supervised optimization could further enhance appearance-structure consistency.

4.4.3.4 Explore Multi-View and Temporal Data Fusion

Current single-frame inference limits robustness in robotic applications. Future systems should leverage multi-view images or video sequences, incorporating temporal consistency and spatial constraints to refine pose estimation accuracy.

4.4.3.5 Enhance Data Diversity with Simulation-to-Reality Adaptation

Despite synthetic data complexity, domain gaps persist between virtual and real factory environments. Subsequent efforts should apply domain adaptation techniques (e.g., domain randomization, style transfer) to minimize distribution discrepancies and improve generalization.

5
Conclusion

This work presents a systematic evaluation of two 6D pose estimation frameworks, YOLO-6D and SAM-6D, within the context of industrial object manipulation. While the YOLO-6D architecture offers superior computational efficiency, its performance is adversely affected by geometric distortions commonly encountered in real-world factory settings and its limited capability in handling intricate part geometries. In contrast, SAM-6D partially mitigates these challenges through the incorporation of zero-shot segmentation and hierarchical geometry scoring, achieving an accuracy of 97% on synthetic datasets that simulate industrial environments and 91% on datasets representing complex, real-world conditions. The SAM-6D framework effectively integrates semantic alignment, appearance consistency, and geometric compatibility—three critical components for achieving robust pose estimation under occlusion and sensor noise.

The evaluation further highlights key limitations, including memory constraints in large-scale point cloud matching and discrepancies between synthetic and real sensor data. Future research should prioritize optimization of memory usage for high-complexity components and the integration of realistic sensor noise models during training. Additionally, aligning the 6D pose estimation module with motion control systems on the ROS platform constitutes an essential direction for practical deployment.

In conclusion, SAM-6D represents a significant advancement in zero-shot 6D pose estimation, offering a scalable and adaptable solution for industrial automation applications where data scarcity and object variability remain central challenges. By narrowing the gap between synthetic training environments and real-world deployment, this work lays a foundation for the development of more adaptive and resilient robotic systems in manufacturing and related domains.

5.0.1 Division of Work

In this project, Yu Kang is responsible for generating datasets using Blenderproc2, modeling with Blender, and collecting data on objects in the factory environment. Junjie Hu is responsible for training YOLO-6D and SAM-6D models, as well as writing and optimizing algorithms.

Bibliography

[1] Y. Sun, S. Dai, J. Dang, and J. Yong, "A 6D object pose estimation method combining self-attention mechanism," 2024 5th International Conference on Computer Engineering and Application (ICCEA), pp. 1315–1319, Apr. 2024. doi:10.1109/iccea62105.2024.10604264

[2] R. Newbury et al., "Deep learning approaches to grasp synthesis: A Review," IEEE Transactions on Robotics, vol. 39, no. 5, pp. 3994–4015, Oct. 2023. doi:10.1109/tro.2023.3280597

[3] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6D pose estimation," International Journal of Computer Vision, vol. 128, no. 3, pp. 657–678, Nov. 2019. doi:10.1007/s11263-019-01250-9

[4] J. Lin, L. Liu, D. Lu, and K. Jia, "Sam-6D: Segment anything model meets zero-shot 6D object pose estimation," 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 27906–27916, Jun. 2024. doi:10.1109/cvpr52733.2024.02636

[5] L. Yang, X. Cui, F. Qin, R. Chen, and F. Li, "BSPNet: A neural network for 6D object pose estimation based on improved yolo-6d," International Conference on Artificial Intelligence, Automation and High Performance Computing, pp. 443–448, Jul. 2024. doi:10.1145/3690931.3691005

[6] L. Zhang et al., "A posture detection method for augmented reality–aided assembly based on yolo-6d," The International Journal of Advanced Manufacturing Technology, vol. 125, no. 7–8, pp. 3385–3399, Jan. 2023. doi:10.1007/s00170-023-10964-7

[7] J. Kang, W. Liu, W. Tu, and L. Yang, "Yolo-6d+: Single shot 6d pose estimation using privileged silhouette information," 2020 International Conference on Image Processing and Robotics (ICIP), pp. 1–6, Mar. 2020. doi:10.1109/icip48927.2020.9367354

[8] Y. Chen, X. Xiong, H. Fang, and Y. Xu, "Ba-sam: Boundary-aware adaptation of segment anything model for Medical Image segmentation," 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 3115–3118, Dec. 2024. doi:10.1109/bibm62325.2024.10822711

[9] S. S. Roy, R. Kardan, and J. Neubert, "Melseg: An adaptation of segment anything model for skin lesion segmentation," 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC), pp. 00347–00351, Jan. 2025. doi:10.1109/ccwc62904.2025.10903695

[10] B. Xu et al., "Multidimensional exploration of segment anything model for weakly supervised video salient object detection," IEEE Transactions on Cir-

cuits and Systems for Video Technology, vol. 35, no. 4, pp. 2987–2998, Apr. 2025. doi:10.1109/tcsvt.2024.3368053

[11] E. Su, H. Cao, and A. Knoll, "BISEG-sam: Weakly-supervised post-processing framework for boosting binary segmentation in segment anything models," 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2430–2437, Dec. 2024. doi:10.1109/bibm62325.2024.10822087

[12] H.-J. Hwang, J.-H. Cho, and Y.-T. Kim, "Deep learning-based real-time 6d pose estimation and multi-mode tracking algorithms for citrus-harvesting robots," Machines, vol. 12, no. 9, p. 642, Sep. 2024. doi:10.3390/machines12090642

[13] J. Wang et al., "Improving angular estimation using a deep CNN network in 6D pose estimation," 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 53–58, Aug. 2022. doi:10.1109/mipr54900.2022.00017

[14] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "POSECNN: A convolutional neural network for 6d object pose estimation in cluttered scenes," Robotics: Science and Systems XIV, Jun. 2018. doi:10.15607/rss.2018.xiv.019

[15] A. S. Periyasamy, C. Capellen, M. Schwarz, and S. Behnke, "Convposecnn2: Prediction and refinement of dense 6d object pose," Communications in Computer and Information Science, pp. 353–371, 2022. doi:10.1007/978-3-030-94893-1_16

[16] 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," 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3848–3856, Oct. 2017. doi:10.1109/iccv.2017.413

[17] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DOF pose estimation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4556–4565, Jun. 2019. doi:10.1109/cvpr.2019.00469

[18] D. Maji, S. Nagori, M. Mathew, and D. Poddar, "Yolo-6D-pose: Enhancing yolo for single-stage monocular multi-object 6D pose estimation," 2024 International Conference on 3D Vision (3DV), pp. 1616–1625, Mar. 2024. doi:10.1109/3dv62453.2024.00160

[19] Z. Luo, J.-T. Hsieh, L. Jiang, J. C. Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged modalities," Lecture Notes in Computer Science, pp. 174–192, 2018. doi:10.1007/978-3-030-01264-9_11

[20] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," Neural Networks, vol. 22, no. 5–6, pp. 544–557, Jul. 2009. doi:10.1016/j.neunet.2009.06.042

[21] S. S. Roy, R. Kardan, and J. Neubert, "Melseg: An adaptation of segment anything model for skin lesion segmentation," 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC), pp. 00347–00351, Jan. 2025. doi:10.1109/ccwc62904.2025.10903695

[22] Y. Chen, X. Xiong, H. Fang, and Y. Xu, "Ba-sam: Boundary-aware adaptation of segment anything model for Medical Image segmentation," 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 3115–3118, Dec. 2024. doi:10.1109/bibm62325.2024.10822711

[23] M. Shahraki, A. Elamin, and A. El-Rabbany, “SAMNet++: A segment anything model for supervised 3D point cloud semantic segmentation,” Remote Sensing, vol. 17, no. 7, p. 1256, Apr. 2025. doi:10.3390/rs17071256

[24] V. Batista, V. Biazi, A. C. Rodrigues, A. M. Soares, and C. Marques, “Fast segment anything model applied in a low-cost camera-based system for Oyster Growth Monitoring,” AIP Advances, vol. 14, no. 12, Dec. 2024. doi:10.1063/5.0233879

[25] J. Wang, W. Liu, C. Xu, and M. Song, “Segmentation, tracking, and safety operation monitoring of tower cranes based on MobileSAM and ByteTrack,” 2024 43rd Chinese Control Conference (CCC), pp. 6881–6886, Jul. 2024. doi:10.23919/ccc63176.2024.10661524

[26] Y. Liu et al., “Gen6D: Generalizable Model-free 6-DOF object pose estimation from RGB images,” Lecture Notes in Computer Science, pp. 298–315, 2022. doi:10.1007/978-3-031-19824-3_18

[27] J. Sun et al., “OnePose: One-shot object pose estimation without CAD models,” 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6815–6824, Jun. 2022. doi:10.1109/cvpr52688.2022.00670

[28] J. Chen et al., “ZeroPose: Cad-prompted zero-shot object 6d pose estimation in cluttered scenes,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 35, no. 2, pp. 1251–1264, Feb. 2025. doi:10.1109/tcsvt.2024.3482439

A

Appendix 1

A.1 Simulation dataset generation

A.1.1 Command to download background image

```
blenderproc download cc_textures
```

A.1.2 Modify the number of object references in other datasets

```
sampled_target_bop_objs = list(np.random.choice(target_bop_objs, size=1,
    replace=False))
sampled_distractor_bop_objs = list(np.random.choice(tless_dist_bop_objs,
    size=1, replace=False))
sampled_distractor_bop_objs += list(np.random.choice(ycbv_dist_bop_objs,
    size=1, replace=False))
sampled_distractor_bop_objs += list(np.random.choice(tyol_dist_bop_objs,
    size=1, replace=False))
```

A.1.3 Command to generate dataset

```
blenderproc run examples/datasets/bop_challenge/main_lm_upright.py ./
    ./backgrounds ./output
```

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY