robotics

MDPI

Article

A Lightweight Vision-Guided Framework for Multi-Object Robotic Grasping and Polynomial Spatial Mapping

Siyi Wan and Mohammad El-Yabroudi *

Department of Electrical and Computer Engineering, Lawrence Technological University, Southfield, MI 48075, USA; swan1@ltu.edu
* Correspondence: melyabrou@ltu.edu

Abstract

Robotic grasping based on monocular vision remains challenging due to perception uncertainty, pixel-to-world mapping errors, and a tight per-frame budget. This paper presents a lightweight vision-based multi-object robotic grasping framework that integrates object detection, polynomial pixel-to-world mapping, and rule-based grasp execution into a unified pipeline. Object detection is carried out with a You Only Look Once version 13 (YOLOv13) detector, into which lightweight structural modules are inserted for the tabletop grasping task. To enable accurate perception-to-action mapping, a polynomial calibration model is introduced to transform image-space detections into planar world-space coordinates, effectively correcting geometric distortions caused by preprocessing. Based on the mapped targets, a rule-based grasp planning strategy is developed to generate physically feasible grasp poses for a Universal Robots 5 (UR5) robotic arm without relying on depth sensors or learning-based grasp prediction. The proposed framework is implemented and validated in CoppeliaSim. Quantitative results show end-to-end grasp success rising from 86% with the center-point baseline to 94% with the proposed strategy on the same 50 paired placements, while mapping and planning together add about 1 ms per frame. The gain comes from enforcing execution feasibility on an already sufficient map, not from a more accurate mapping model.

Keywords: monocular vision; robotic grasping; pixel-to-world mapping; YOLO-based detection; rule-based grasp planning

Check for updates

Academic Editors: Konstantinos Tsintotas, Ioannis Kostavelis and Kyriakos Tsiakmakis

Received: 5 September 2026
Revised: 8 October 2026
Accepted: 13 October 2026
Published: 16 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Robotic grasping is a fundamental and challenging problem in autonomous manipulation, especially in unstructured environments where control over object poses, illumination, and spatial context is difficult to enforce [1]. Vision-based techniques let robots sense and interact with their environment, which adds flexibility in industrial and service applications [2,3]. Among available sensing modalities, monocular vision is especially attractive because of its low cost, simple setup, ease of deployment, and scalability. However, reliable robotic grasping under monocular vision remains an open problem [4,5].

Unlike conventional automated production lines that depend on pre-programmed trajectories and precisely engineered fixtures, vision-based systems enable robots to locate, recognize, and grasp objects with varying shapes, poses, and lighting conditions [6,7]. Conventional grasping relies heavily on object models and analytical computation of grasp poses, whereas detection-based methods use convolutional neural networks (CNNs) to infer graspable regions directly from visual input [8,9]. This transition motivates the use of

learning-based perception modules in robotic grasping pipelines. Among existing detection frameworks, single-stage detectors such as You Only Look Once (YOLO) are especially suitable for robotic vision because they perform object localization and category prediction in a single forward pass [10]. This design provides a favorable balance between detection accuracy and inference speed, which is what keeps the detector inside the 33 ms frame budget of Equation (22).

Recent advances in deep learning have greatly improved object detection accuracy, enabling reliable localization of multiple objects in complex scenes [11,12]. However, high detection accuracy does not necessarily translate into accurate grasp execution. For monocular systems, grasp success is directly dependent on the transformation from image coordinates to robot workspace coordinates [13]. Small geometric discrepancies introduced during image preprocessing, camera calibration, or coordinate mapping can accumulate along the perception–planning–execution pipeline and may result in grasp failure. This issue is often insufficiently addressed in existing studies, which tend to emphasize perception performance rather than full-system geometric consistency. These geometric errors are not isolated. Detection uncertainty, the transformation between image and workspace, and the execution constraints of the arm all act on the same final quantity, which makes monocular grasping a system-level problem rather than a detection task. Instead of adding new sensors or relying on learning-based grasp prediction networks, this work focuses on improving grasp execution accuracy through explicit geometric modeling and system integration. A lightweight pipeline is constructed that incorporates object detection, polynomial pixel-to-world mapping, and rule-based grasp planning within a closed-loop control framework. These components are well established in the literature, and the contribution of this work lies in how they are connected rather than in the components themselves.

The contributions of our work are summarized as follows:

1. A system-level integration of monocular detection, polynomial pixel-to-world mapping, and rule-based grasp execution is described, in which the geometric conventions established at one stage are preserved through the next. The individual modules are adopted from existing methods, and in this work the interface conventions and consistency conditions are specified.
2. The propagation of errors from detection and mapping to the final grasp result is quantified, using test samples that are independent of the data used to fit each stage. This allows the end-to-end result to be attributed to the individual stages rather than to the pipeline as a whole.
3. The complete pipeline is evaluated in a simulated tabletop setting with a Universal Robots 5 (UR5) arm, and per-category mapping accuracy on independent test points, and end-to-end grasp success are reported, so that the effect of geometric consistency can be examined separately from that of detection performance.

2. Related Work

Vision-based robotic grasping sits at the interface between perception and motion. The earliest strategies were geometric. They tested force closure on a known object model. Inside a structured cell, they worked. Each one needed the same three things: an accurate three-dimensional (3D) model, stable conditions, and a calibration that held. Occlusion, a reflective surface, a deformable object, or a target placed somewhere new was enough to break it. A feasible grasp was defined by contact points, surface normals, and friction constraints. Wan et al. [6] proposed a visual detection strategy for rough castings from 3D vision measurements, but its robustness rests on precise modeling and surface calibration. Yin et al. [7] traced robotic grasp detection from two-dimensional (2D) to 3D perception

and noted the limited adaptability of geometric methods under complex objects and scenes. That limit pushed the field toward data-driven and learning-based grasping.

Learning-based grasping estimates grasp poses directly from visual input and has largely replaced analytical pipelines for unknown objects. CNNs learn hierarchical visual representations, so they are now standard for object detection, grasp pose estimation, and affordance prediction. They handle texture and lighting variation far better than hand-crafted features do. A large part of this literature is RGB-D: depth plus a six-dimensional (6D) pose, learned end-to-end, with high success rates on cluttered tables. The cost is structural. Such methods need a depth camera and a large annotated dataset. Both raise calibration effort and compute. Chen et al. [14] proposed a vision-based grasping method built on deep reinforcement learning (DRL). Repeated simulated interactions refine its policy until it generalizes to unseen objects under closed-loop control. Qi et al. [15] took the same route toward human-like dexterous grasping, adding multimodal perception. Two problems persist. The learned representations are hard to interpret, which is a poor fit for safety-critical use. Additionally, the training itself is expensive. Moving a simulation-trained policy onto a real robot is a further step, and the sim-to-real gap [16] is what makes that step hard.

Industrial and service scenes put several objects in front of the robot at once. The task is no longer to grasp one isolated object; it is to choose which one, and to do so without disturbing the others. YOLO-based detectors fit this setting well. They localize objects quickly, and their outputs feed straight into a grasp prediction module. Du et al. [17] modified the YOLOv3 feature pyramid for multi-object grasp detection and added spatial separation constraints to keep nearby targets distinct in a cluttered scene. Li et al. [18] used large-kernel convolution, residual connections, and feature fusion to strengthen the spatial representation available to a grasp head.

Monocular grasping removes the depth channel. That makes it cheaper and simpler than RGB-D or multi-view setups, and it leaves a geometric problem behind. Song et al. [19] built a hierarchical RGB-D fusion model that reads depth boundaries together with color texture. It predicts grasp rectangles. With a single camera there is nothing to fuse. The relationship between what the image shows and where the robot must move has to be inferred. Two families of methods exist. Some pair monocular perception with heuristic grasp planning, others recover the missing dimension from pose estimation or a second view. Both remain exposed to the same thing: the mapping between image coordinates and workspace coordinates. A few pixels of detection error become a positional error in the workspace, and the gripper follows the mapping, not the image.

Across these lines the same trade-off recurs. Geometric methods are accurate but brittle [6,7]; RGB-D and learned methods are flexible, but they need hardware and data that the task does not always justify [14,19]. The gap is narrower. Detection accuracy is reported on its own, grasp prediction accuracy on its own. Rarely are the two reported together with the mapping and the execution that connect them. This work takes the whole chain as its unit of study: lightweight YOLOv13-based detection, a polynomial pixel-to-world mapping, rule-based grasp planning, and closed-loop execution on a UR5. There is no depth camera and no learned grasp network. The question is whether the chain holds end to end.

3. Methodology

3.1. System Overview

The overall setup of the proposed monocular robotic grasping system is depicted in Figure 1. The system integrates visual perception, geometric mapping, motion planning, and robotic execution into a unified pipeline.

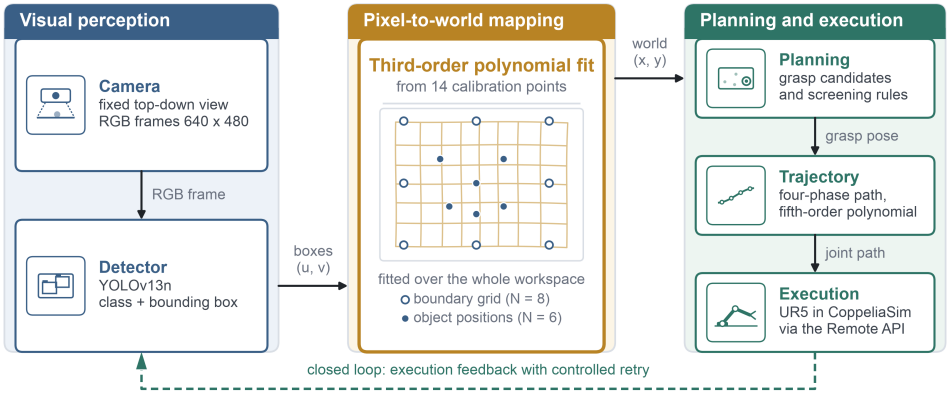


Figure 1. Overall pipeline of the proposed monocular robotic grasping system, which couples a YOLOv13n detector, a polynomial pixel-to-world mapping, and a rule-based grasp planner driving a UR5 arm, and returns execution feedback with controlled retry in a closed loop.

An RGB image captured by a fixed monocular camera is first processed by a YOLOv13-based object detector to determine the category and centroid of the bounding box in the image space. The transformation from perception to action first applies inverse geometric reconstruction to compensate for image preprocessing effects, and then uses polynomial pixel-to-world mapping to transform image coordinates into planar world coordinates [16,20]. Based on the transformed targets, a rule-based grasp planner finds grasp poses and trajectories feasible for the UR5 robotic arm. The system operates in a closed-loop manner. Execution feedback from the simulator is fed continuously back to the system, implementing controlled retry in case of detection or execution failure. First, the robotic operations are constrained to a planar workspace observed by a fixed top-down monocular camera. Second, a unified right-handed coordinate system with metric units is maintained across perception, mapping, planning, and execution modules. Third, image preprocessing operations, including resizing and letterbox padding, are recorded and consistently reversed before pixel-to-world mapping. Finally, the evaluation in this work is conducted in CoppeliaSim [21], where the influence of perception, mapping, and execution errors can be isolated under controlled conditions. A tilted or moving camera would change the projection between image space and the workspace, so the calibration would have to be re-estimated for each viewpoint. Stacked or deformable objects break the premise that the detected center lies in the grasp plane. Full six-degree-of-freedom grasp planning would require the orientation to be optimized rather than fixed along the workspace normal. None of these extensions are tested in this work.

3.2. Visual Perception Module

3.2.1. YOLOv13-Based Detection Backbone

Object detection is implemented using a YOLOv13-based single-stage detector [10]. The single-stage formulation is what keeps the per-frame detection cost small enough for the closed loop of Section 3.5. The detector outputs object classes, confidence scores, and bounding-box coordinates in the network input coordinate system. In this work, the YOLOv13 perception module serves as the baseline, on which lightweight structural modules are added. Instead of changing the overall detection paradigm, this work concentrates on lightweight structural augmentations that provide localization robustness while staying inside the 33 ms per-frame budget of Section 3.5. The visual perception module is based on convolutional feature extraction. As illustrated in Figure 2, convolution and pooling operations progressively transform the input image into hierarchical feature representations,

which are then used for object classification and localization. This feature extraction process provides the basis for the YOLOv13-based detector used in the proposed grasping pipeline.

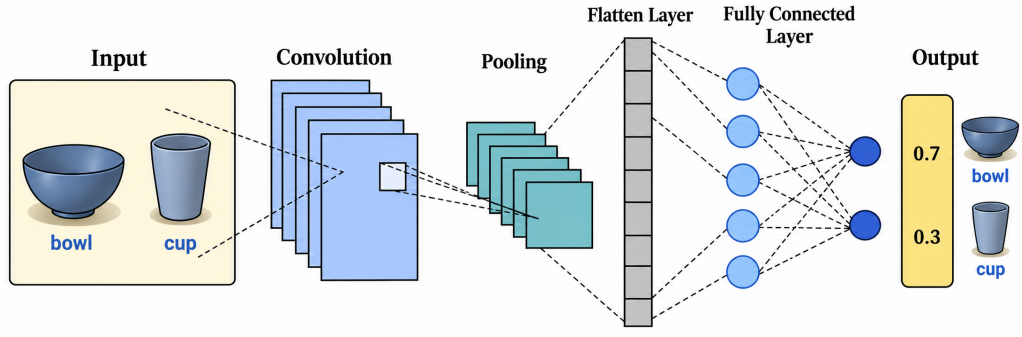


Figure 2. Schematic structure of CNN-based visual feature extraction for object detection.

Given the original image size (W_0, H_0) and the target input size imgsz, the scaling factor used during preprocessing is defined as

$$s = \min\left(\frac{\text{imgsz}}{W_0}, \frac{\text{imgsz}}{H_0}\right). \tag{1}$$

where W_0 and H_0 denote the width and height of the original image, respectively, and imgsz denotes the square network input size. The symmetric padding values are computed as

$$p_x = \frac{\text{imgsz} - sW_0}{2}, \quad p_y = \frac{\text{imgsz} - sH_0}{2}. \tag{2}$$

The detector output is uniformly represented in the network input coordinate system as

$$\mathbf{y} = \{\text{class}, (c'_x, c'_y, w', h'), \text{score}\}. \tag{3}$$

where (c'_x, c'_y) denotes the bounding-box center in the network input coordinate system, (w', h') denotes its width and height, and the prime symbol indicates coordinates after network preprocessing. During inference, the parameters (s, p_x, p_y) are recorded together with the detection result so that the original pixel coordinates can be reconstructed before coordinate mapping.

3.2.2. Lightweight Structural Enhancements

To improve detection of small objects and objects that are partially occluded, three lightweight enhancement units are introduced into the detection backbone: Coordinate Attention (CA) and Spatial Pyramid Pooling (SPP), both inserted into the native YOLOv13 architecture, together with a controlled variant of the detection head used for the head-design ablation described in Section 4.4. CA module: The CA module improves spatial feature representation by embedding positional information into channel attention. It then pools globally (both height and width) and allows long-range dependencies, which benefits the localization accuracy of the objects. This is particularly beneficial for occluded objects or objects with high background similarity. Given an input feature map $X \in \mathbb{R}^{C\times H\times W}$, the output feature map Y is computed as follows [22]:

$$Y_c(i,j) = X_c(i,j) \times g^h(i) \times g^w(j) \tag{4}$$

where g^h and g^w denote attention weights along height and width, respectively; c indexes the feature channel; (i, j) denotes the spatial location; and Y is the attention-refined

output feature map. This enhances the spatial sensitivity and localization accuracy in a cluttered scene.

SPP is employed to aggregate multi-scale contextual information using parallel max-pooling operations with different kernel sizes [23]. The feature is expressed as:

$$f_{\text{spp}} = \text{Concat}\Big[\text{MaxPool}_{k_1\times k_1}(X), \text{MaxPool}_{k_2\times k_2}(X), \text{MaxPool}_{k_3\times k_3}(X)\Big] \tag{5}$$

where k_1, k_2, and k_3 denote pooling kernel sizes at different receptive-field scales. This structure is also robust to scale variations in the evaluated dataset, without a substantial computational overhead. The decoupled head is designed to separate the classification and regression jobs into two independent branches to reduce task conflict during training. The total loss is given by [24]:

$$L_{\text{total}} = \lambda_1 L_{\text{cls}} + \lambda_2 L_{\text{reg}} + \lambda_3 L_{\text{obj}} \tag{6}$$

where L_{cls}, L_{reg}, and L_{obj} denote classification, regression of the bounding box, and the objectness loss, respectively, and λ_1, λ_2, and λ_3 are their balance weights.

The enhancement modules are introduced as pluggable units at component-level without changing the overall YOLO detection paradigm. CA is inserted into the middle- and high-level backbone stages to strengthen spatial localization. SPP is placed before the neck to aggregate multi-scale contextual features. The native YOLOv13 head is already decoupled, with separate regression and classification branches; Section 4.4 treats this design as a controlled factor rather than as an added module. This design improves localization stability while preserving the per-frame latency advantage of the single-stage detector. The corresponding insertion positions of these lightweight enhancement modules are summarized in Figure 3.

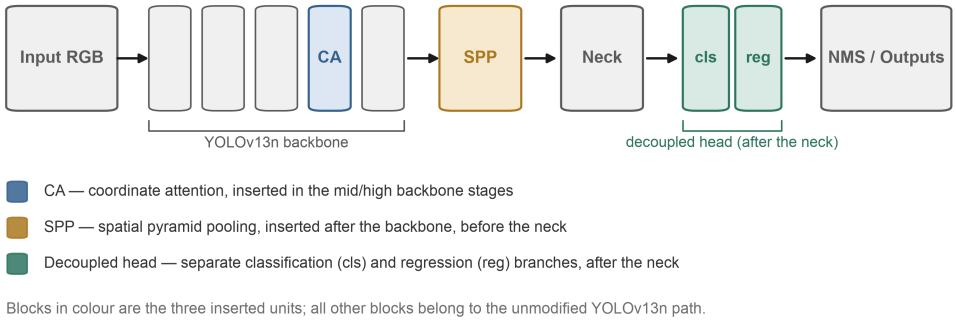


Figure 3. Insertion positions of the lightweight enhancement modules in the YOLOv13-based perception module.

3.2.3. Output Consistency and Inference Settings

Geometrically, to achieve consistency between perception and execution, all inference outputs follow the same coordinate convention. Image resizing and letterbox padding parameters recorded during inference allow for inverse geometric reconstruction at subsequent inference of pixel-to-world mapping. After the detection module outputs bounding-box centers, widths, and heights in the network output format, the module provides reliable inputs for downstream geometric transformation and grasp planning.

3.3. Pixel-to-World Coordinate Mapping

3.3.1. Inverse Geometric Reconstruction

During inference, input images are uniformly resized and letterbox-padded. Let the pixel coordinates predicted by the detection network be denoted as (u', v'). The scaling factor and padding parameters are represented as (s, p_x, p_y). Given a cropping offset $(\Delta x, \Delta y)$, which is zero when no cropping is applied, the original image coordinates (u, v) follow as:

$$u = \frac{u' - p_x}{s} + \Delta x, \quad v = \frac{v' - p_y}{s} + \Delta y \tag{7}$$

The first stage reverses the resizing and letterbox padding applied during network inference, recovering the original pixel coordinate (u, v) from the network coordinate (u', v'). The second stage maps the reconstructed pixel coordinate to the planar world coordinate (x, y) through a calibrated polynomial transformation. The complete coordinate conversion process is illustrated in Figure 4.

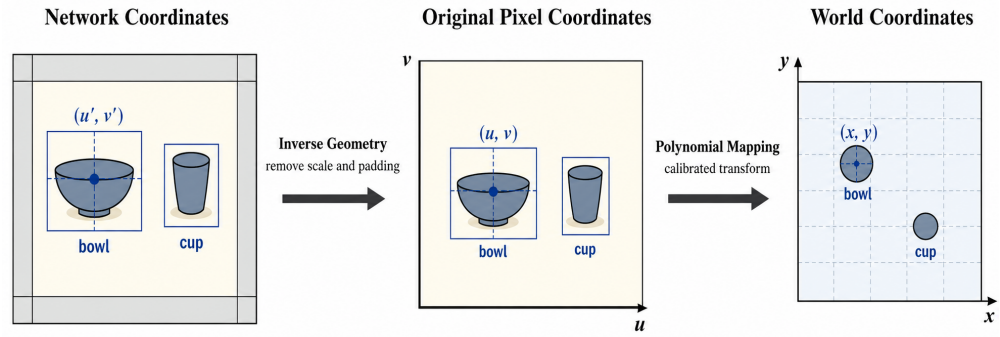


Figure 4. Two-stage coordinate conversion from network image coordinates to planar world coordinates.

These inverse conversions apply uniformly to bounding-box centers, bounding-box corners and other key points of a box, so the same mapping can be applied to any of them. The camera is a perspective sensor, and over the fronto-parallel mapping plane its ideal pinhole model reduces to a fixed scale factor f/D, where D is the camera-to-plane distance; the polynomial in Section 3.3.2 therefore only has to absorb the residual geometric error rather than the projection itself. For the parameters used here ($f = 554.26$ px, $D = 1.20$ m) the implied scale is 2.17 mm px^{-1}, which the calibration pairs reproduce to within 0.5%.

3.3.2. Polynomial Mapping Model

After inverse geometric reconstruction, pixel coordinates (u, v) are transformed to planar world coordinates (x, y) using a polynomial. A third-order polynomial (poly3) mapping model is tested to investigate the influence of polynomial models on mapping accuracy. The poly3 model is defined as:

$$\begin{aligned} x &= a_0 + a_1 u + a_2 v + a_3 u^2 + a_4 uv + a_5 v^2 \\ &\quad + a_6 u^3 + a_7 u^2 v + a_8 u v^2 + a_9 v^3 \\ y &= b_0 + b_1 u + b_2 v + b_3 u^2 + b_4 uv + b_5 v^2 \\ &\quad + b_6 u^3 + b_7 u^2 v + b_8 u v^2 + b_9 v^3 \end{aligned} \tag{8}$$

where $\{a_i, b_i\}$ are the polynomial coefficients to be determined. For comparison, a second-order polynomial (poly2) mapping model can be written as:

$$x = a_0 + a_1 u + a_2 v + a_3 u^2 + a_4 uv + a_5 v^2, \tag{9}$$

$$y = b_0 + b_1 u + b_2 v + b_3 u^2 + b_4 uv + b_5 v^2. \tag{10}$$

Between the two polynomial models, the third-order one is adopted because the calibration budget supports it: its ten coefficients per axis are fewer than the fourteen calibration points used in this work, so that the least-squares fit remains over-determined, whereas a fourth-order model would require fifteen and could not be identified from the same set. The accuracy of this model against affine, homography, and second-order mappings is compared, under a common protocol, in Section 4.6.

3.3.3. Calibration and Coefficient Estimation

During calibration, correspondence pairs $\{(u_i, v_i) \rightarrow (x_i, y_i)\}$ are collected from the simulation environment. In this work $N = 14$ points were recorded, chosen to cover the workspace geometrically rather than randomly: eight form a regular grid on the boundary of the calibrated region ($x \in \{0.6, 1.0, 1.4\}$ m, $y \in \{-0.4, 0.0, 0.4\}$ m) and six lie at the object positions actually used in the grasp experiments. The polynomial coefficients are then estimated by least squares, minimizing the coordinate mapping error:

$$(c_x^*, c_y^*) = \arg\min_{c_x, c_y} \sum_{i=1}^{N} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right], \tag{11}$$

where $(\hat{x}_i, \hat{y}_i)$ represent the predicted world coordinates obtained from the polynomial model. The calibration samples form a linear least-squares problem. For the third-order model, the polynomial basis is defined as $\phi(u, v) = [1, u, v, u^2, uv, v^2, u^3, u^2v, uv^2, v^3]$. Using this basis, the mapping can be expressed as:

$$\hat{x} = Ac_x, \quad \hat{y} = Ac_y, \tag{12}$$

where A is the Vandermonde matrix constructed from pixel-coordinate basis terms, and c_x and c_y denote the coefficient vectors for the two world-coordinate axes. For N calibration points, $A \in \mathbb{R}^{N \times 10}$ for the third-order polynomial basis. The coefficients are solved by minimizing the reprojection error over all calibration points. Because the same points cannot serve to both estimate and validate the mapping, all mapping errors reported in Section 4.6 are obtained under a leave-one-out protocol: for each of the $N = 14$ points, the coefficients are re-estimated from the remaining 13 points and the held-out point is used only for evaluation. Calibration and evaluation data are therefore disjoint at every reported point, and no point contributes to the model that predicts it. The number of basis terms is given in Section 4.6, and in every case it is smaller than the number of points retained for fitting, so all 14 points can be evaluated without discarding any.

3.4. Grasp Planning and Execution Strategy

3.4.1. Grasp Candidate Generation

From the pixel-to-world mapping results, grasp candidates are derived under a top-down grasping assumption, which is appropriate to planar manipulation problems that typically occur in industrial settings. For each detected object, the center of the mapped bounding box is the grasp target on the workspace plane. The grasp pose is defined by a fixed end-effector orientation along the workspace normal and a given approach direction. This definition avoids orientation optimization and improves grasp reliability in the planar setting studied here. Additionally, candidate grasp positions are restricted by geometric workspace limits to ensure the reachability of the robot arm. To ensure feasibility, each candidate must satisfy workspace and gripper constraints. Let $\Omega \subset \mathbb{R}^2$ denote the valid planar workspace. A candidate (x, y) is accepted only if

$$(x, y) \in \Omega, \quad x_{min} \leq x \leq x_{max}, \quad y_{min} \leq y \leq y_{max}. \tag{13}$$

where $\Omega = [0.65, 1.35] \times [-0.35, 0.35]$ m is the 0.70×0.70 m planar workspace of Table 1, and the synthetic dataset of samples object centers inside Ω, inset by the object radius plus a 0.03 m clearance so that a sampled object always lies fully inside the workspace. The estimated object width $\hat{w}$, recovered from the mapped points of the detected object, must also be compatible with the gripper opening range:

$$g_{min} \leq \kappa \hat{w} \leq g_{max}, \tag{14}$$

where $g_{\min}$ and $g_{\max}$ are the admissible gripper openings ($g_{\min} = 0.020$ m and $g_{\max} = 0.110$ m, the stroke of the gripper listed in Table 1), and $\kappa = 0.75$ sets the commanded opening below the measured width so that the fingers contact the object before the joint limit. Every target in this dataset satisfies Equation (14); the constraint is stated because it bounds the set of objects the planner can accept.

Table 1. Camera and simulation parameters used throughout the experiments. The camera is an ideal pinhole sensor with no lens distortion; the focal length follows from the field of view and the resolution, $f = (W/2)/\tan(\mathrm{FOV}/2)$. The ground sampling distance is evaluated on the mapping plane at $z = 0.70$ m, i.e., 1.20 m below the camera.

Group	Parameter	Value
Camera	Projection model	Perspective (ideal pinhole, no distortion)
	Native resolution	640×480 px
	Horizontal field of view	60.0°
	Focal length	$f = 554.26$ px
	Principal point	$(c_x, c_y) = (320.0, 240.0)$ px
	Near/far clipping	0.01 m/10.0 m
	Camera pose (world frame)	(1.000, 0.000, 1.900) m
	Optical axis	World $-z$ (fixed top-down view)
	Height above plane	1.20 m
	Ground sampling distance	2.17 mm px^{-1}
Simulation	Simulator	CoppeliaSim 4.10
	Physics engine	Bullet 2.78
	Simulation step	0.05 s
	Gravity	$(0, 0, -9.81)$ m s^{-2}
Manipulator	Arm	UR5, base at $(0, 0, 0.700)$ m
	End effector	OnRobot RG2 two-finger parallel gripper
	Gripper opening range	$g_{\min} = 0.020$ m, $g_{\max} = 0.110$ m (Equation (14))
Workspace	Area	0.70×0.70 m at $z = 0.70$ m

3.4.2. Geometric and Workspace Constraints

In addition to planar reachability, clearance and height constraints are applied during execution. The approach height is set above the workspace plane to avoid premature contact, while the grasp height is defined near the object surface according to the top-down grasping assumption. Category-dependent safety offsets can be introduced when necessary to prevent the gripper from colliding with object edges or the workspace boundary.

3.4.3. Four-Phase Trajectory Planning

Each valid grasp candidate is executed using a four-phase trajectory including approach motion, grasp motion, lift motion and retreat motion. The approach motion moves the end-effector from above a safe height to a specified pre-grasp position above the target. Grasp phase moves the gripper to the grasp height and closes the gripper. Lift phase moves the gripper up from grasp height to the safe height and the retreat phase moves the end

effector out of workspace. Each phase is generated using smooth polynomial interpolation to maintain continuity of position, velocity, and acceleration. Inverse kinematics solutions are computed by the simulation environment, and only collision-free solutions are accepted for execution. The sampled path and the four phases are shown in Figure 5 together with the fifth-order polynomial that is used inside each phase.

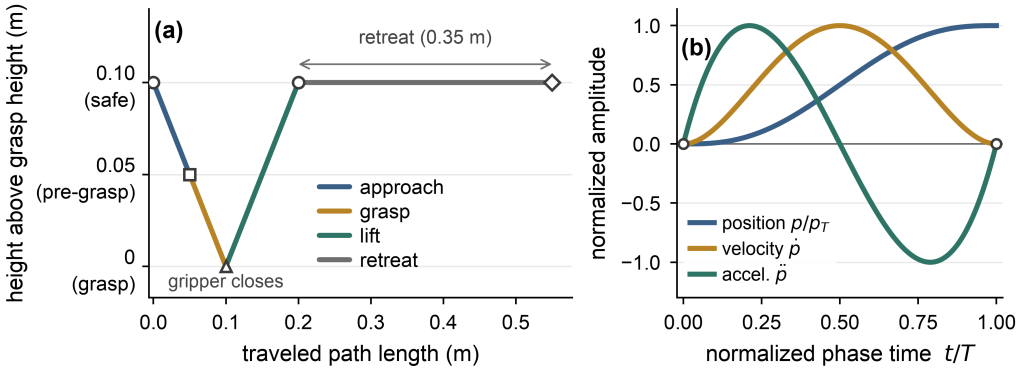


Figure 5. Four-phase trajectory planning strategy for top-down grasp execution. (**a**) Height of the end effector along the sampled path, measured relative to the grasp height; (**b**) The fifth-order polynomial of Equation (15) over one phase, normalized. Position, velocity and acceleration all vanish at the phase boundaries, which is what makes the phase transitions continuous.

For each Cartesian component or joint variable $p(t)$, a fifth-order polynomial trajectory is used:

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \tag{15}$$

The boundary conditions are defined as

$$p(0) = p_0, \quad \dot{p}(0) = 0, \quad \ddot{p}(0) = 0, \tag{16}$$

$$p(T) = p_T, \quad \dot{p}(T) = 0, \quad \ddot{p}(T) = 0. \tag{17}$$

Under these constraints, the coefficients are

$$a_0 = p_0, \quad a_1 = 0, \quad a_2 = 0, \tag{18}$$

$$a_3 = \frac{10\Delta p}{T^3}, \quad a_4 = -\frac{15\Delta p}{T^4}, \quad a_5 = \frac{6\Delta p}{T^5}, \tag{19}$$

where p_0 and p_T are the initial and terminal positions of each trajectory segment, and $\Delta p = p_T - p_0$. This ensures continuous position, velocity, and acceleration between trajectory segments.

3.4.4. Execution Strategy and Failure Handling

The execution module monitors each grasp attempt and triggers a bounded retry mechanism when timeout or positioning errors occur. During execution, the simulator returns the end-effector state and grasp outcome, which are used to determine whether the current target is completed, retried, or skipped. The target position is perturbed within a small radius before replanning:

$$(x, y) \leftarrow (x, y) + \delta(\cos\theta, \sin\theta). \tag{20}$$

where θ is sampled from $[0, 2\pi]$ and δ is limited by a predefined maximum perturbation. The number of retries is bounded to prevent the system from being trapped by a single failed target. Our execution strategy avoids replanning and improves reliability under

the perception uncertainty characterized in Section 4.6. Together with explicit geometric constraints, the bounded retry mechanism improves grasp reliability without introducing significant additional computational overhead.

3.5. System Timing and Reproducibility

The cost of one placement separates into two quantities that are owned by different parts of the system. The perception-to-planning chain contains the stages the vision pipeline owns, namely image capture, detection, pixel-to-world mapping, and grasp selection:

$$\tau_{\text{chain}} = \tau_{\text{capture}} + \tau_{\text{det}} + \tau_{\text{map}} + \tau_{\text{plan}}, \tag{21}$$

where τ_{capture} is the cost of reading one frame from the sensor and transferring it to the host, excluding the inter-frame wait, and τ_{det}, τ_{map} and τ_{plan} denote detection, coordinate mapping, and grasp selection. The grasp cycle $\tau_{\text{cycle}} = \tau_{\text{chain}} + \tau_{\text{exec}}$ additionally contains the manipulator trajectory, which is governed by the robot controller rather than by the vision pipeline and is three orders of magnitude larger. Only the chain is subject to a timing requirement. Writing T_{budget} for the frame period of the nominal control rate, the system satisfies that requirement when

$$E[\tau_{\text{chain}}] \leq T_{\text{budget}}, \tag{22}$$

where $E[\cdot]$ denotes the expected value over processed frames. Throughout this paper the nominal control rate is 30 Hz, so $T_{\text{budget}} = 33$ ms. The measured stage costs are reported in Section 5.2.

4. Experimental Setup and Evaluation

4.1. Simulation Environment and Experimental Setup

All experiments are conducted in the CoppeliaSim simulation environment using a UR5 robotic manipulator equipped with a parallel gripper. A fixed monocular RGB camera is placed above the workspace area providing top view. It is implemented as an ideal pinhole vision sensor with a native resolution of 640×480 px and a horizontal field of view of 60.0°, corresponding to an equivalent focal length of $f = 554.26$ px and a ground sampling distance of 2.17 mm px^{-1} on the mapping plane at $z = 0.70$ m; no lens distortion is modelled. The camera is mounted eye-to-hand and kept in this top-down configuration in every experiment, so its pose, $(1.000,\ 0.000,\ 1.900)$ m with the optical axis along the world $-z$ direction, is identical throughout. Camera and simulation parameters are listed in Table 1.

The workspace is modeled as a planar surface, and all target objects are placed within the predefined reachable region of the robot. The software and hardware configuration of the experiment is summarized in Table 2.

Table 2. Software and hardware environment configuration.

Category	Component/Setting	Description
Software	Python 3.12/PyTorch 2.8.0/Ultralytics 8.3.228	System integration, deep learning framework, and YOLOv13 detection
	CoppeliaSim + ZMQ Remote API	Scene generation, image capture, and robotic control
Hardware	NVIDIA GeForce RTX 4090 D (24 GB)	Training of every detector configuration and all latency measurements
	Apple Silicon (MPS)	Pipeline development and offline debugging only
	CPU/RAM	8 cores, 32 GB memory

Note: The training and evaluation protocol is given in Section 4.4.

Cups and bowls were selected as target objects because they are common in tabletop manipulation tasks and have different grasping geometries. In each experiment, the objects were randomly placed within the workspace with different positions and different orientations. The robot then performs grasping attempts based on visual perception outputs without using depth information. For fair comparison, the camera parameters, robot configuration, workspace settings, and execution strategy are kept identical across all experiments. As shown in Figure 6, the robot, camera, workspace, and target objects are arranged within the simulated manipulation scene.

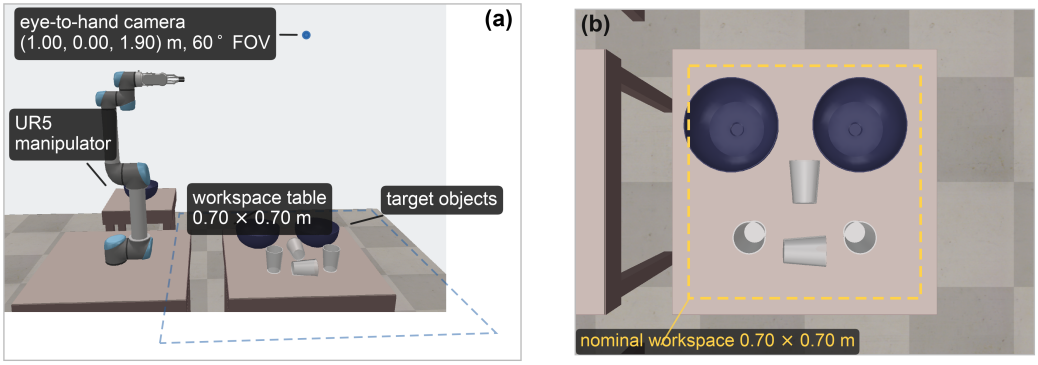


Figure 6. Simulation environment and experimental setup in CoppeliaSim. (**a**) Oblique view of the cell: the UR5 manipulator on its own table and the 0.70 × 0.70 m workspace table carrying the target objects. The eye-to-hand camera is marked by the dot; the dashed square is the footprint of its field of view on the mapping plane at $z = 0.70$ m, which is wider than the panel and therefore runs off the lower edge. (**b**) The top-down view of the eye-to-hand camera; the dashed rectangle marks the nominal 0.70 × 0.70 m workspace.

The 500 images span two object categories and were generated under controlled randomization, so that an end-to-end failure can be attributed to detection, mapping or execution rather than to dataset shift. Generalization to further categories and to real imagery is outside the scope of this study and is listed in Section 6.

4.2. Automated Dataset Generation Pipeline

To improve perception robustness under varying visual conditions, an automated synthetic dataset generation pipeline was developed in CoppeliaSim. The pipeline generates training samples without manual image annotation by using the known object poses and scene parameters available in the simulation environment. Each data collection cycle consists of four main stages: scene initialization, parameter randomization, image capture, and automatic annotation.

In the scene initialization stage, the UR5 manipulator, tabletop workspace, target objects, virtual camera, and our fixed non-directional light sources are loaded into the simulation scene. The target objects include cup and bowl models, which are selected because they represent common tabletop grasping targets with different shapes and grasping geometries. In the parameter randomization stage, object positions, object orientations, lighting intensity, lighting direction, and visual appearance are randomly varied within predefined workspace limits. This randomization increases the visual diversity of the dataset and helps the detector adapt to different illumination and pose conditions.

During image capture, the virtual monocular camera records RGB images from the workspace. Since the simulator provides access to ground-truth object positions and scene states, the corresponding object labels can be generated automatically. The automatic annotation module projects the visible object information into the image plane and exports the labels in YOLO format, including the object class and normalized bounding-box param-

eters. The pipeline can produce image–label pairs efficiently while avoiding the cost and inconsistency of manual labeling.

The overall workflow of the automated dataset generation process is shown in Figure 7. Starting from the simulated cup and bowl models, the scene is randomized, captured by the virtual vision sensor, and converted into a labeled dataset for training the detection model.

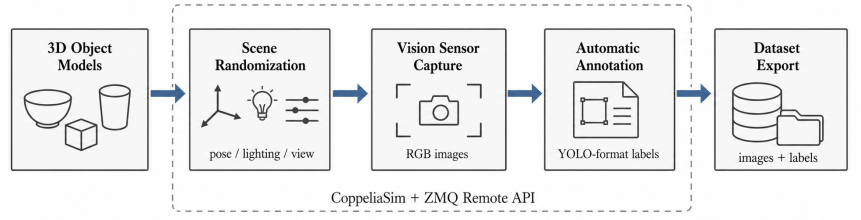


Figure 7. Automated synthetic dataset generation pipeline in CoppeliaSim.

The detailed dataset composition and randomization settings are summarized in Table 3. A total of 500 synthetic images were generated, covering two target categories: cups and bowls. The object positions were sampled within the predefined planar workspace, while lighting conditions and object poses were randomized to increase visual diversity. These settings allow the dataset to cover common variations in tabletop robotic grasping, including changes in object location, orientation, illumination intensity, and local shadow distribution.

Table 3. Composition of the Synthetic Dataset Generated in CoppeliaSim

Item	Value/Range	Description
Total images	500	Captured in a single automated batch.
Object classes	cup, bowl	Two target categories rendered in the workspace.
Object size	cup: ∅ 88.2 mm × 123.3 mm bowl: ∅ 258.5 mm × 105.3 mm	Outer diameter and height of each object model, measured on the mesh bounding box.
Instances per image	2	One cup and one bowl are included in each scene.
Instances per class	cup: 500; bowl: 500	Equal counts for the two classes.
Workspace (x, y)	$[0.65, 1.35] \times [-0.35, 0.35]$ m	The 0.70 × 0.70 m planar workspace of Table 1; the accepted region Ω of Equation (13).
Object center (x, y)	bowl $[0.85, 1.15] \times [-0.15, 0.15]$ m cup $[0.76, 1.24] \times [-0.24, 0.24]$ m	The sampling region is inset from the workspace by a per-class margin (0.17 m bowl, 0.08 m cup) plus a 0.03 m clearance.
Lights on ratio	≈75%	Each light is activated with probability 0.75.
Light intensity	0.35–1.0	Randomly scaled relative intensity.
Light RGB tint	0.85–1.0 × intensity	Random per-channel color variation.
Light pose jitter	yaw ±25°, pitch ±15°	Orientation perturbation for each light source.
Cup pose distribution	upright ≈ 27%, upside-down ≈ 7%, slightly tilted ≈ 24%, tilted-inverted ≈ 10%, side ≈ 33%	The cup orientation is drawn from five discrete poses.

The balanced category distribution and randomized scene parameters reduce sampling bias and provide diverse visual inputs for training and evaluating the detector.

In addition to the randomization rules, the generated dataset was examined statistically to verify workspace coverage and category balance. As shown in Figure 8a, the generated samples cover the intended planar workspace rather than concentrating in a narrow region, which helps reduce location bias during detector training. Representative synthetic images are shown in Figure 8b. These examples demonstrate variations in illumination intensity, local shadows, object orientation, and object placement. Such variations approximate common visual disturbances in tabletop grasping scenes while maintaining full control over ground-truth object positions and YOLO-format labels in simulation.

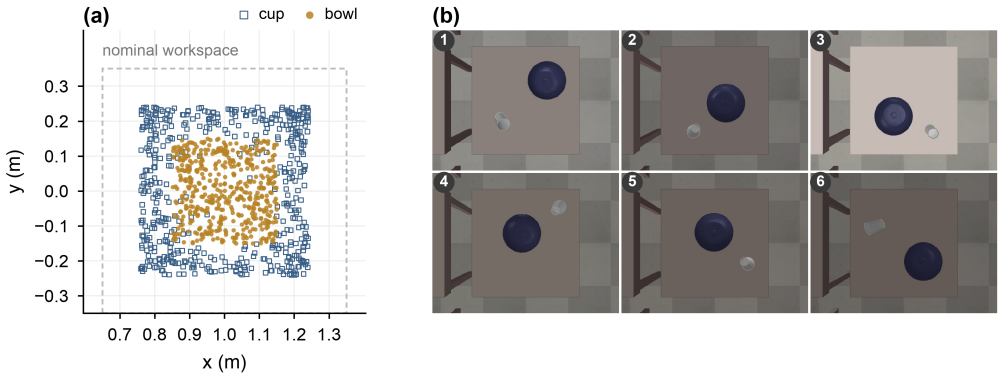


Figure 8. Generated synthetic dataset. (**a**) The 500 sampled object placements; the dashed square marks the nominal 0.70 × 0.70 m workspace. (**b**) Six representative captures of the set, shown at their native 640 × 480 px resolution.

4.3. Evaluation Metrics

The dataset generation process is designed to evaluate robustness under perception uncertainty caused by changes in object pose, lighting, and camera observation. In each capture, the object positions, object orientations, and illumination were randomized, while the camera remained fixed in the top-down configuration of Section 4.1. The detector is therefore evaluated on held-out object poses and lighting conditions, but not on held-out camera viewpoints; generalization across viewpoints would require re-estimating the calibration for each new projection. The 500 captures were split at the image level into 400 training, 50 validation and 50 test images, and the three splits are disjoint (Table 4). The detection metrics of Section 4.5 are reported on the 50 test images, so the detector is scored on scenes it has not been trained on. We use three types of evaluation to measure the different points along the perception–execution pipeline:

1. Detection performance measured by mean Average Precision (mAP) and inference speed.
2. Pixel-to-world mapping accuracy evaluated by the planar localization error between predicted and ground-truth world coordinates.
3. End-to-end grasp success rate, defined as the ratio of successful grasps to total grasp attempts.

These metrics collectively provide insight into how perception accuracy and geometric consistency affect execution reliability. To make the evaluation criteria explicit, the quantitative metrics used in this work are defined as follows. The planar pixel-to-world mapping error is computed as:

$$e_i = \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}, \tag{23}$$

where (x_i, y_i) denotes the ground-truth world coordinate of the i-th reference point, and $(\hat{x}_i, \hat{y}_i)$ denotes the corresponding coordinate predicted by the calibrated mapping model.

Table 4. Training and Evaluation Protocol.

Item	Setting
Input image size	640 × 640 pixels (letterboxed from the native 640 × 480 capture)
Object categories	cup and bowl
Training epochs	200 (fixed for every configuration; early stopping disabled)
Random seeds	0, 1, 2 per configuration ($n = 3$)
Dataset split	400/50/50 images (train/validation/test), disjoint at the image level
Learning rate schedule	Cosine decay
Momentum	0.937
Weight decay	5×10^{-4}
Inference batch size	1 (per-frame timing)
Post-processing	Non-maximum suppression included
Evaluation metrics	mAP@0.5:0.95, box-center error, detection latency, mapping RMSE, grasp success rate
Downstream detector	YOLOv13n baseline (see Section 5.2)

The overall mapping accuracy is reported using the root mean square error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}, \tag{24}$$

where N is the number of evaluated reference points. Throughout this paper e_i denotes the planar error of a single reference point (Equation (23)) and RMSE denotes the aggregate over all N evaluated points. The two are reported separately and are not interchangeable.

For grasp execution, the success rate is defined as:

$$S = \frac{N_s}{N_t} \times 100\%, \tag{25}$$

where N_s denotes the number of successful grasp attempts and N_t denotes the total number of grasp attempts. A grasp is counted as successful only when the robot lifts the target object from the workspace without collision or slipping.

4.4. Training Protocol and Ablation Design

The object detection model was trained using the synthetic dataset generated in the CoppeliaSim environment. The detailed training and evaluation settings are summarized in Table 4. These settings are kept consistent across all YOLOv13-based variants to ensure that the comparison reflects the effect of the structural modifications rather than differences in training configuration.

The detection task contains two object categories, namely cups and bowls. All annotations follow the YOLO format defined in the dataset configuration file, where each label contains the class index and normalized bounding-box parameters. The input image size was set to 640 × 640 pixels for both training and inference to maintain consistency between the detection model and the downstream coordinate mapping module.

All YOLOv13-based variants were trained under the same protocol to ensure a fair comparison. Each model was trained for 200 epochs using a cosine-decay learning rate schedule, with momentum set to 0.937 and weight decay set to 5×10^{-4}. Unless otherwise specified, identical data preprocessing, augmentation, training, and evaluation settings were used for all variants. Inference speed was measured with batch size equal to 1, including non-maximum suppression (NMS), on the same computing device.

To characterize the detector that feeds the downstream stages, a full-factorial ablation was conducted on the YOLOv13n baseline. Three binary factors were varied: coordinate attention (CA), spatial pyramid pooling (SPP), and the detection-head design. Because

the native YOLOv13 head is already decoupled, the third factor is realized by replacing it with a coupled head in which regression and classification share a single branch; it is a negative control for the decoupling of the native design rather than an added module. All eight resulting configurations were trained under the protocol of Table 4 and are compared in Table 5. Mapping and grasp experiments from Section 4.6 onward use the YOLOv13n baseline, whose box-center error is representative of all eight configurations (Section 5.2).

Table 5. Full-factorial ablation of the YOLOv13n detector over the three binary factors CA, SPP, and detection-head design.

Variant	mAP@0.5:0.95	Center Err (px)	Latency (ms)	Params (M)	GFLOPs
YOLOv13n with the native (decoupled) head					
YOLOv13n (baseline)	0.944 ± 0.006	1.054 ± 0.019	24.2 ± 1.0	2.46	6.37
+CA	0.943 ± 0.006	1.056 ± 0.039	25.1 ± 3.2	2.51	6.42
+SPP	0.942 ± 0.003	1.086 ± 0.041	23.5 ± 2.2	2.62	6.50
+CA + SPP	0.944 ± 0.003	1.094 ± 0.028	29.7 ± 0.3	2.67	6.55
YOLOv13n with the coupled head (negative control)					
Coupled head	0.944 ± 0.002	1.048 ± 0.014	23.4 ± 1.5	2.52	6.79
Coupled head + CA	0.948 ± 0.003	1.056 ± 0.038	21.0 ± 2.3	2.57	6.84
Coupled head + SPP	0.947 ± 0.006	1.046 ± 0.021	22.2 ± 2.6	2.69	6.92
Coupled head + CA + SPP	0.942 ± 0.005	1.039 ± 0.016	23.6 ± 1.9	2.73	6.98

Note: The native YOLOv13 head is decoupled, with separate regression and classification branches; the third factor replaces it with a coupled head that shares a single branch, and is therefore a negative control for the decoupling rather than an added module. Every value is the mean $\pm$ standard deviation over three seeds (0, 1, 2), each trained for 200 epochs with early stopping disabled. "center err" is the mean Euclidean distance between the predicted and the ground-truth box center in the 640×640 network input frame. Latency is the end-to-end per-frame time at batch size 1 including non-maximum suppression, measured on an NVIDIA GeForce RTX 4090 D shared instance in a single session; it reflects the relative cost of the eight configurations rather than absolute throughput. Params and GFLOPs are computed at 640×640 and are independent of the measurement session.

4.5. Detection Performance Evaluation

Table 5 reports mAP@0.5:0.95 averaged over the three seeds, the mean box-center error, and the per-frame detection latency. The box-center error is what carries over to the downstream stages: it stays between 1.04 and 1.09 px in the 640×640 input frame, so the mapping and planning stages receive box centers of the same quality from every configuration, and all experiments from Section 4.6 onward therefore use the YOLOv13n baseline.

Across the eight configurations the mean mAP@0.5:0.95 lies between 0.942 and 0.948, and the spread between configurations (0.007) does not exceed the seed-to-seed standard deviation of the individual configurations (up to 0.006). Neither the two inserted modules nor the head design therefore separate by a margin this dataset can resolve. What the table establishes is that the configurations are equivalent within measurement noise, not that any of them is better; the contribution claimed for these modules is architectural (they are inserted at the stage boundaries shown in Figure 3 without disrupting the single-stage design) rather than a measured accuracy gain.

4.6. Pixel-to-World Mapping Accuracy

Mapping accuracy is quantified by the planar distance between the predicted and the reference world coordinates, evaluated over the $N = 14$ reference points under the leave-one-out protocol described in Section 3.3.3. Every error reported in this subsection is therefore out-of-sample: each prediction is produced by a model fitted on all calibration points except the one being scored. The corresponding in-sample values are given in the last column of Table 6, so that the two conventions can be told apart.

The six object positions are a subset of the same calibration set, and they are the positions the pipeline has to reach during grasping. Table 7 reports the residual at each of them, with a mean per-point error of 0.92 cm and a worst case of 1.62 cm ($N = 6$). The signs of Δx and Δy vary across the six points instead of pointing the same way, so the residual is not a common displacement in a fixed direction and no translational correction would remove it.

Table 6. Comparison of pixel-to-world mapping models over the 14 calibration points. All columns except the last are leave-one-out errors, and SD is their standard deviation over the 14 points; the last column reports the residual on the same points used for fitting. For the homography, so the basis-terms values are not directly comparable across models. Bold marks the lowest value in each column.

Model	Basis Terms	Mean ± SD (cm)	Median (cm)	P95 (cm)	RMSE (cm)	Max (cm)	In-Sample RMSE (cm)
Affine	3	1.284 ± 0.830	**1.039**	2.788	1.529	3.107	1.280
Homography	8	1.585 ± 0.903	1.901	2.808	1.824	3.123	1.280
Polynomial (2nd)	6	1.859 ± 1.061	2.323	3.085	2.140	3.168	1.258
Polynomial (3rd)	10	**1.109** ± 0.693	1.114	**2.299**	**1.307**	**2.651**	0.365

Table 7. Per-point pixel-to-world mapping error at the object positions.

Point	Reference Position (x, y) (m)	Mapped Position (x, y) (m)	Δx (cm)	Δy (cm)	e_i (cm)
1	(0.800, 0.155)	(0.809, 0.147)	0.85	−0.82	1.18
2	(1.150, 0.155)	(1.148, 0.153)	−0.20	−0.15	0.25
3	(1.000, 0.000)	(0.999, 0.003)	−0.14	0.29	0.32
4	(0.850, −0.150)	(0.848, −0.160)	−0.17	−0.97	0.98
5	(1.000, −0.200)	(1.000, −0.188)	−0.03	1.16	1.16
6	(1.150, −0.150)	(1.157, −0.165)	0.66	−1.47	1.62

Note: mean e_i = 0.92 cm; median = 1.07 cm; maximum = 1.62 cm; RMSE = 1.04 cm (Equation (24), $N = 6$).

Table 6 puts the four models side by side over the full calibration set under the same protocol. The third-order model attains the lowest mean error (1.109 cm), the lowest RMSE (1.307 cm), the lowest 95th percentile (2.299 cm) and the smallest maximum error (2.651 cm, against 3.107 cm for the affine model), while the affine model has the lowest median (1.039 cm against 1.114 cm). The third-order model also has the smallest spread (SD 0.693 cm against 0.830 cm for the affine model), so its lower mean is not the result of a few large residuals. A paired t-test over the 14 held-out points separates the third-order model from the second-order one ($p = 0.035$, better at 10 of the 14 points) but not from the affine model ($p = 0.56$, better at 7 of the 14). The last column shows why the two conventions have to be kept apart: the in-sample RMSE of the third-order model (0.365 cm) is 3.6 times smaller than its out-of-sample value, so a single figure that did not distinguish them would overstate its accuracy by that factor.

The comparison therefore supports selecting the third-order model from among these four, but it does not establish a general advantage of higher-order mappings over lower-order ones, since the margin over the affine model is at the resolution of this calibration set. What it does show is that accuracy at this stage is not gained by adding basis terms. The failure modes that arise from the remaining error are discussed in Section 5.4. Figure 9 shows the mapping output on the reference frame, with the mapped center of each detection marked against the object it belongs to.

Reported by category, the mapping residual differs between the two object classes by a factor of about two. A capture is written in the same simulation step as the placement

that produced it, so each capture can be paired with the placement record of the step that precedes it; the first of the 500 captures has no predecessor and is excluded. Of the 1000 annotated instances, 998 therefore carry a paired placement record (499 cups and 499 bowls), and these are the instances scored here. Over the 998 detections that were not used to fit the mapping, the mean error is 0.928 cm for cups and 0.482 cm for bowls. The ratio is close to two for the third-order model and remains above one for the affine and second-order models as well. The cause is not the mapping itself, which maps a given pixel offset to the same displacement for either class, but the bounding-box center: a cup subtends roughly one sixth of the image area of a bowl, so the same absolute center localization error is a larger fraction of its box, and the randomized cup orientation further varies the box aspect ratio. This also explains why the failure cases analyzed in Section 5.4 concentrate on the smaller object.

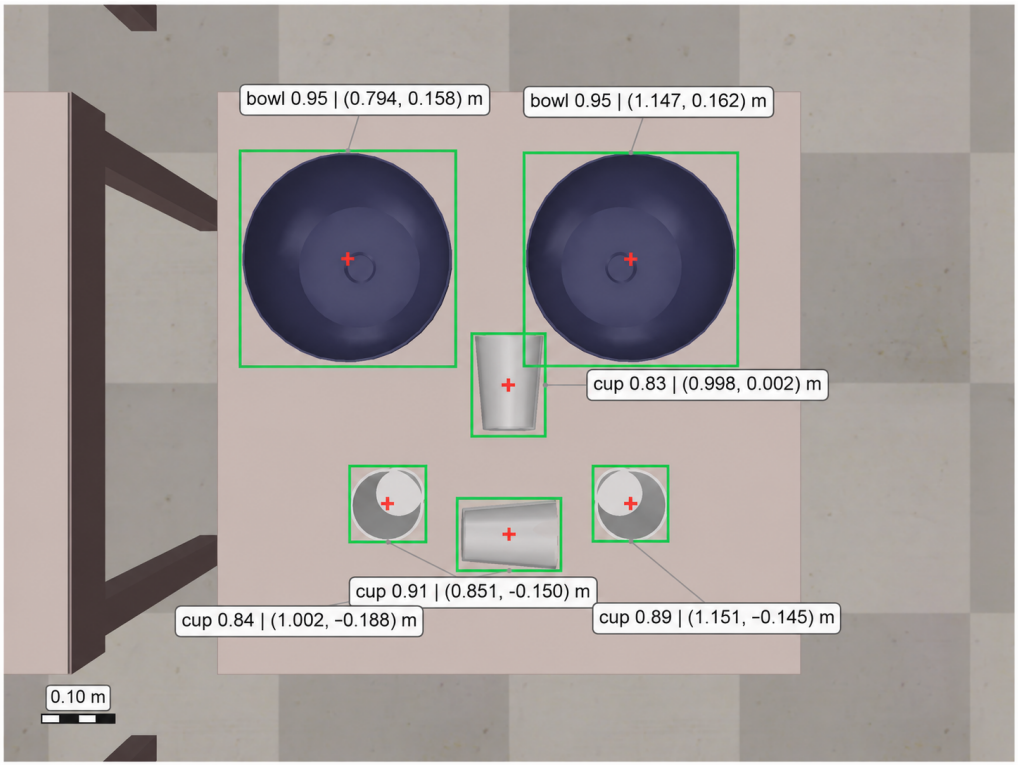


Figure 9. Pixel-to-world mapping results for the reference frame. The six detections of the baseline detector are outlined in green and the mapped bounding-box center of each instance is marked in red; the labels give the object class, the detection confidence and the planar world coordinates obtained with the third-order polynomial mapping. The segmented bar at the lower left spans 0.10 m on the mapping plane.

4.7. End-to-End Grasp Success Evaluation

End-to-end grasp success is evaluated through grasp trials that involve different random object orientations. A grasp is counted as successful under the criterion of Section 4.3 (Equation (25)). Both strategies are scored by the same criterion on the same 50 placements. We compare the proposed grasp planning and execution strategy with a center-point baseline that uses the mapped bounding-box center without further geometric refinement. The two strategies are run on the same 50 target placements, so the trials form a paired design and the success counts can be compared directly (Table 8). The rule-based strategy succeeds on 47 placements and the baseline on 43, a gap of 4 placements out of 50. A difference of this size is a consistent trend rather than a statistically established effect, and we make no claim of significance. Execution time rises by 0.3 s, from 7.5 s to 7.8 s, which is small

next to the cycle time itself and reflects the feasibility checks and bounded retries of the rule-based planner. Per-category success counts are not reported. Splitting the 50 paired placements by class leaves about 25 trials per class, a sample at which a difference of two or three placements cannot be separated from sampling noise; the comparison in Table 8 is therefore stated as a trend rather than an established effect. The class dependence is quantified instead where the sample is large: over the 998 scored detections the mean localization error is 0.928 cm for cups and 0.482 cm for bowls (Section 4.6), which follows from the bounding-box geometry, since a cup subtends about one sixth of the image area of a bowl. The same mechanism is visible in Section 5.4, where the failures concentrate on the smaller object.

Table 8. Comparison between center-point and rule-based grasping strategies.

Method	Trials	Successes	Success Rate (%)	Execution Time (s)
Center-point	50	43	86.0	7.5
Rule-based	50	47	94.0	7.8

Note: Both strategies were evaluated on the same 50 target placements, so the counts form a paired design. Success is scored by the criterion defined in Section 4.3, Equation (25).

The 50 planned poses in Figure 10 are collision-free and mutually consistent for the same multi-object scene. All arrows represent grasp vectors in transformed world coordinates, showing both the grasp point and the approach direction; they are drawn at the same on-page length, so they indicate direction only.

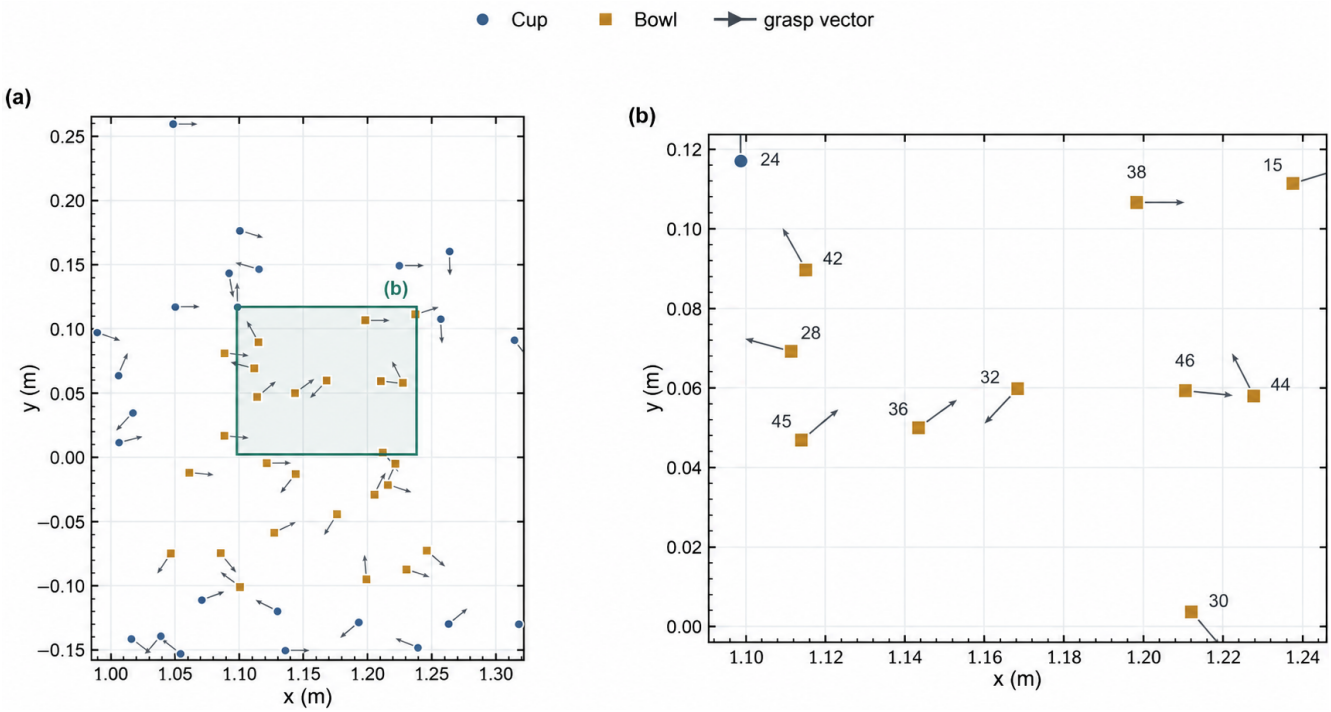


Figure 10. Rule-based grasp planning results in world coordinates. (**a**) All 50 planned grasp targets, color- and shape-coded by object class; the boxed region is enlarged in (**b**). (**b**) Magnification of the boxed region in (**a**); the numbers are the target indices and each arrow gives the planned approach direction. All arrows are drawn at the same on-page length and therefore indicate direction only. Cup targets are shown as circles and bowl targets as squares, which keeps the two classes distinguishable in greyscale print.

5. Results and Discussion

5.1. Comparative Analysis with Existing Studies

Recent studies on vision-based robotic grasping have mainly focused on improving object detection accuracy, grasp pose prediction, or end-to-end grasping performance. However, these approaches differ significantly in sensing modality, network architecture, dataset source, and system-level integration. A direct numerical comparison is difficult because the experimental platforms, object categories, and evaluation protocols are not identical. Instead, this section provides a qualitative comparison to clarify the design focus and practical characteristics of the proposed framework.

Representative methods include YOLO-based multi-object grasp detection [17], RGB-D fusion grasp prediction [19], and convolutional grasping models with enhanced spatial representation [18]. Sun et al. proposed a collaborative robot grasping system based on a parallel YOLO and grasp-generation architecture, where YOLO was used for object localization and a grasp prediction network was used to estimate grasp poses [9]. These studies demonstrate that deep visual perception and grasp prediction can improve robotic manipulation performance. However, many existing methods either rely on additional sensing hardware, such as depth cameras, or focus mainly on isolated perception or grasp prediction modules rather than the complete perception-to-execution chain.

To better summarize the differences between representative studies and the proposed method, Table 9 compares several vision-based robotic grasping approaches in terms of core architecture, dataset source, grasp prediction strategy, system-level validation, and main limitations. The difference Table 9 makes visible is narrow, the listed methods reach comparable success rates while adding hardware, a second view, or a learned predictor.

Table 9. Comparison With Representative Vision-Based Robotic Grasping Studies

Study	Method Characteristics	Validation Setting	Main Limitation
Sun et al. [9]	YOLO-based object localization combined with a grasp-generation network; learning-based grasp prediction.	Real-robot validation with task-specific data.	Requires two-network integration and introduces synchronization complexity.
Du et al. [17]	Improved YOLOv3 for multi-object grasp detection with feature-pyramid refinement and spatial constraints.	Simulation or dataset-level evaluation using a custom multi-object dataset.	Focuses mainly on grasp detection rather than complete perception-to-execution validation.
Song et al. [19]	Hierarchical RGB-D fusion using depth boundary information and color texture cues for grasp prediction.	Dataset-level validation with RGB-D data.	Requires depth sensing and additional calibration.
Li et al. [18]	Large-kernel convolution and residual learning for enhanced spatial representation in robotic grasping.	Experimental or public-data validation with learning-based prediction.	Provides limited analysis of pixel-to-world mapping error and execution robustness.
This work	YOLOv13-based monocular detection combined with polynomial pixel-to-world mapping and rule-based grasp planning.	CoppeliaSim simulation using automatically generated synthetic data.	Physical robot validation is not included in the current stage.

As shown in Table 9, existing approaches often improve either perception accuracy or grasp pose prediction, whereas the proposed framework focuses on the consistency of the complete perception–mapping–planning–execution pipeline. The proposed method does not require depth sensing or an additional learning-based grasp prediction network, which reduces system complexity and computational cost. Meanwhile, the explicit polynomial mapping module makes it possible to analyze how coordinate transformation errors affect grasp reliability. The main limitation is the simulation setting; Section 6 lists what that leaves untested. Nevertheless, the controlled simulation environment allows the influence of detection accuracy, pixel-to-world mapping, and execution constraints to be isolated and evaluated systematically.

5.2. Detection Performance and System Bottleneck Analysis

The stage costs of the perception-to-planning chain defined in Section 3.5 are reported in Table 10.

Table 10. System-level summary of the proposed framework.

Stage	Metric	Value
Detection (YOLOv13n baseline, 640 × 640, batch 1)	mAP@0.5:0.95 (mean of three seeds)	0.944 ± 0.006
	Box-center error	1.05 ± 0.02 px
	τ_{det}, incl. preprocessing and NMS	24.2 ± 1.0 ms
Pixel-to-world mapping (third-order polynomial)	RMSE (leave-one-out, $N = 14$)	1.31 cm
	Maximum per-point error ($N = 14$)	2.65 cm
	τ_{map}	0.14 ± 0.04 ms
Grasp selection (rule-based)	τ_{plan}	0.78 ± 0.15 ms
Perception-to-planning chain against a frame budget of $T_{\text{budget}} = 33$ ms (Equation (22))	τ_{chain} (Equation (21))	25.1 ms
Grasp execution (same 50 paired placements)	Success rate, center-point baseline	43/50 (86%)
	Success rate, proposed rule-based	47/50 (94%)

Note: τ_{det} is the baseline value of Table 5 measured on an NVIDIA GeForce RTX 4090 D shared instance; τ_{map} and τ_{plan} are pure geometric operations executed in single-threaded NumPy and are accelerator-independent. The optional interface overlay used during interactive runs costs about 19 ms but is not part of the deployed chain and is excluded from Equation (22). The two success rates come from the same 50 placements and differ by a net of four (Section 4.7).

Detection dominates the chain by more than an order of magnitude, 24.2 ms against 0.92 ms for mapping and grasp selection combined. The chain is detection-bound, and at this workspace scale the remaining stages are not candidates for further optimization. The measured chain, 25.1 ms, meets the 33 ms budget of Equation (22) with margin, whereas a single grasp cycle takes 7.5 s; the cycle time is therefore set by the manipulator and not by the detector. As shown in Table 5, all YOLOv13-based variants reach nearly saturated detection accuracy, with a mean mAP@0.5:0.95 between 0.942 and 0.948 and a per-frame detection latency between 21 and 30 ms at batch size 1 on the NVIDIA GeForce RTX 4090 D used for all timing measurements. These comparisons suggest that detection accuracy, as measured by mAP, is already saturated on this dataset, and that further architectural changes to the detector are unlikely to improve this metric. What these comparisons do not establish is where the remaining error of the complete pipeline originates: that is a question about the localization of the predicted box centers rather than about classification accuracy, and Section 5.4 discusses the failure modes to which this localization question gives rise in practice.

5.3. Impact of Pixel-to-World Mapping Accuracy

The mapping results obtained under the leave-one-out protocol show that localization errors vary across workspace positions. The eight boundary grid points in the calibration set have a mean residual of 1.25 cm against 0.92 cm at the six object positions, so a weak tendency towards larger errors away from the workspace center is present, but with this sample size the difference is not statistically significant (Welch t-test, $p = 0.39$); the correlation between distance from the workspace center and residual is $r = 0.33$ ($p = 0.25$). We therefore treat the boundary effect as a trend observed in this dataset rather than as an established property of the mapping model.

Under the leave-one-out protocol the third-order model reaches an RMSE of 1.31 cm and the affine model 1.53 cm (Table 6). Both remain inside the tolerance that a top-down two-finger grasp imposes. That tolerance is lateral: the grasp still completes as long as the grasp point stays inside the object footprint, so the admissible deviation from the object center is half the object width at the grasp height. For the cup used in this scene, whose outer diameter is 90 mm, the allowance is 45 mm. The leave-one-out RMSE of 1.31 cm uses 29% of that allowance, and the worst single point, 2.65 cm, uses 59%. The mapping model used here is therefore not the factor that limits grasp accuracy at this workspace scale. This does, however, establish something the pipeline depends on: the mapping residual is dominated by point-to-point scatter rather than by a single large directional error, and the coordinate conventions linking image space to the robot base are consistent, which is what allows the residual to be attributed downstream. Section 5.4 illustrates the failure modes that arise when the mapping is not consistent.

To examine the spatial structure of the mapping error, leave-one-out residuals were computed over the full calibration set and are plotted in Figure 11. With $N = 14$ residuals ranging from 0.13 to 2.65 cm, no dominant direction is visible. The largest single residual occurs at a workspace corner, but the four corner points are not uniformly bad (2.65, 2.11, 0.55 and 0.13 cm), which argues against a smooth radial trend and points instead to isolated sensitivity at specific corners. We report the distribution as measured and do not attribute it to a specific distortion mechanism.

The residual analysis indicates that the mapping error is not only determined by the object detector, but also by the geometric consistency between the image coordinate system and the robot workspace. In monocular grasping, even when object detection is nearly saturated, small coordinate deviations may affect the contact position between the gripper and the object. Explicitly modeling the pixel-to-world transformation is necessary for improving end-to-end grasp reliability. Among the models compared here, the third-order polynomial gives the lowest error on independent points, but its advantage over an affine mapping is small and statistically insignificant, and its use depends on a calibration set dense enough to constrain all ten coefficients per axis. The practical implication is the opposite of what a higher-order model is usually chosen for: at this workspace scale the mapping residual is not the term worth optimizing further, because the models compared here already agree within the noise of the calibration set. There is nevertheless a reason to keep the third-order polynomial, and it is not accuracy. The polynomial absorbs the perspective projection and the residual of the image preprocessing together, so it can be re-estimated from calibration pairs alone whenever the camera or the mapping height changes; the affine model cannot play that role, because it presupposes that the image-to-workspace scale is already supplied by the calibration. The ten coefficients per axis are the largest basis the fourteen-point budget identifies without becoming under-determined. The model is retained as a transferable geometric layer, not as a source of lower error.

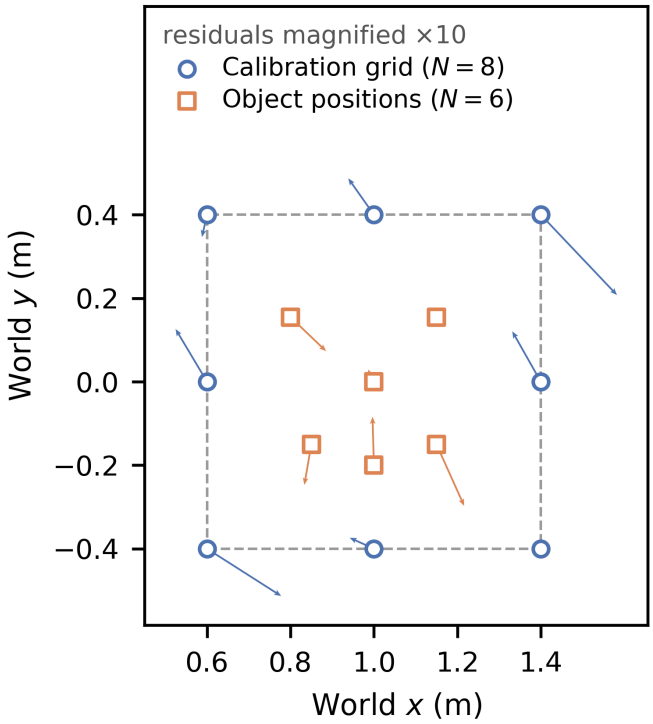


Figure 11. Leave-one-out residuals of the pixel-to-world mapping over the workspace ($N = 14$: eight boundary grid points and six object positions), magnified $\times 10$. The dashed rectangle marks the calibrated workspace region.

5.4. End-to-End Grasp Performance and Failure Modes

End-to-end, the rule-based strategy succeeds on 47 of the 50 placements against 43 for the center-point baseline (Section 4.7). The two strategies share the same mapper and differ only in how the mapped target is turned into a grasp, so the four extra successes are attributable to execution rather than to perception. At this sample size the gap is a trend rather than an established effect, and Section 4.7 states it as such.

Three of the fifty placements fail. Each one is traced below to a stage of the chain, and in every case the error is small enough to be harmless on its own: detection, pixel-to-world mapping, and execution each contribute a term. Since the proposed system relies on monocular vision, the detected 2D object center must be accurately transformed into the planar workspace coordinate. When the detected bounding-box center deviates from the actual object center, the corresponding world coordinate also shifts, which may cause the gripper to approach an incorrect grasping position.

According to the residual analysis of the pixel-to-world mapping model, most mapping errors remain inside the lateral tolerance of 45 mm defined by the cup footprint (Section 5.3). The leave-one-out RMSE over the calibration set is 1.31 cm. The largest single residual occurs at a workspace corner, and whenever a planar error of this size arises, the transformed grasping point may fall slightly away from the physical center of the object even if the object is correctly detected by the visual perception module. This problem is especially important for small objects or objects with narrow graspable regions, because a small planar error can directly reduce grasp stability.

Another common source of failure is inaccurate bounding-box localization under challenging visual conditions. When the object is partially occluded, strongly affected by shadows, or visually similar to the background, the detector may generate a bounding box whose center does not accurately represent the true grasping center. Reflective surfaces or low-contrast regions may reduce the confidence and localization precision of the detector. These failures do not originate in the detector. On the test set it localizes box centers

to within about one pixel (Table 5), so the degradation is confined to localization under occlusion and low contrast.

Execution-related failures were also observed when the planned grasp pose was close to the workspace boundary or when the gripper approached the object with insufficient clearance. In such cases, the manipulator may reach a kinematically feasible position, but the gripper-object alignment is not accurate enough to complete a stable grasp. The retry mechanism alleviates this problem by applying bounded perturbations to the target position and replanning the grasp attempt. However, the retry strategy is simple and does not perform global replanning or object re-identification. Repeated failures may still occur if the original perception or mapping error is too large.

The failure cases show that monocular grasping is not determined by detection accuracy alone. The residual mapping error, the bounding-box localization uncertainty, workspace constraints and the gripper alignment jointly determine the outcome, which is why the pipeline is analyzed as a whole rather than module by module.

6. Conclusions and Future Work

The system described in this paper turns a fixed top-down monocular image into an executed grasp through four stages: YOLOv13-based detection, polynomial pixel-to-world mapping, rule-based grasp planning, and closed-loop execution. Its single premise is that accurate image-space detection does not by itself yield a reliable world-space grasp point. Instead of relying on depth sensors or computationally expensive learning-based grasp prediction networks, the framework emphasizes explicit geometric consistency, efficient visual perception, and physically feasible rule-based execution.

High detection accuracy alone is not sufficient to guarantee successful robotic grasping. Although the YOLOv13-based detector achieved near-saturated detection performance, the end-to-end grasping performance was strongly influenced by the accuracy of the coordinate transformation between the camera image and the robot workspace. With inverse geometric reconstruction and polynomial pixel-to-world mapping, the placement error at the fourteen calibration points is 1.31 cm RMSE under the leave-one-out protocol (Table 6), and the choice of mapping model is not what determines it: the models compared here agree within the noise of the calibration set, which is why refining the mapping further does not change the grasp outcome. A consistent mapping with no large residual shift is therefore a prerequisite for the downstream stages, especially when depth information is unavailable.

The proposed rule-based grasp planning and execution strategy further improved grasp reliability by enforcing workspace constraints, gripper feasibility, safe approach motion, and bounded retry handling. In simulation experiments using a UR5 robotic arm in CoppeliaSim, the proposed framework improved grasp success from 86% using the center-point baseline to 94%, a difference that 50 paired trials do not establish statistically (Section 4.7). This improvement was achieved with only a small increase in execution time, showing that, within the simulated tabletop setting evaluated here, lightweight geometric planning improves grasp success at a small timing cost. These results support the use of low-cost monocular vision for structured tabletop grasping tasks when calibration, mapping, and execution constraints are carefully modeled.

The stage-wise attribution supports several system-level observations. Detection accuracy is already saturated on this dataset and no longer separates the configurations tested here, whereas the residual of the mapping stage is dominated by scatter rather than by a directional shift and stays inside the lateral tolerance of the object class, so an affine model is not measurably worse than the third-order one on independent points. Synthetic dataset generation, in turn, produced the balanced and diverse training data the detector needed while keeping exact ground-truth labels available. Since small errors in each of

these stages accumulate into the end-to-end outcome, the pipeline is best evaluated as a whole rather than module by module.

Despite these promising results, several limitations remain. The current validation was conducted in a simulated environment, where camera noise, contact dynamics, lighting variation, object material properties, and robot calibration errors are more controlled than in physical settings. In addition, the current framework assumes a fixed top-down monocular camera and a planar workspace, which simplifies the pixel-to-world mapping problem. The rule-based grasp planner also uses a fixed top-down grasping strategy and does not yet consider more complex object geometries, deformable objects, severe occlusion, or full six-degree-of-freedom grasp pose optimization.

Future work will focus on extending the proposed framework from simulation to physical robot experiments using a real UR5 manipulator and monocular camera setup. This will allow the effects of real camera calibration errors, mechanical tolerances, lighting disturbances, and contact dynamics to be evaluated. Future studies will also investigate adaptive calibration strategies and online geometric refinement methods to update the pixel-to-world mapping model during operation. In addition, more advanced grasp selection strategies may be incorporated to handle irregular object shapes, cluttered scenes, and partial occlusions while preserving the lightweight nature of the system. Expanding the synthetic dataset with additional object categories, material appearances, and real-world domain randomization will also be explored to improve detection robustness under illumination and material variation. Within the planar, fixed-camera setting evaluated here, the perception-to-planning chain stays well inside the frame budget and the cycle time is set by the manipulator rather than by perception.

Author Contributions: Conceptualization, S.W.; methodology, S.W. and M.E.-Y.; investigation, S.W.; resources, S.W. and M.E.-Y.; data curation, S.W.; writing—original draft preparation, S.W.; writing—review and editing, M.E.-Y. and S.W.; visualization, S.W.; supervision, M.E.-Y.; project administration, M.E.-Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset generated during this study is available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (OpenAI, GPT-5.6) for language editing, grammar correction, improving clarity and readability, and assisting with the refinement of responses to reviewer comments. The authors carefully reviewed and edited all AI-generated output and take full responsibility for the accuracy, integrity, and content of this publication.

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

References

1. Zheng, Y.; Yao, L.; Su, Y.; Zhang, Y.; Wang, Y.; Zhao, S.; Zhang, Y.; Chau, L.P. A survey of embodied learning for object-centric robotic manipulation. *Mach. Intell. Res.* **2025**, *22*, 588–626. [CrossRef]
2. Gao, H.; Zhao, J.; Hu, J.; Sun, C. A real-time grasping detection network architecture for various grasping scenarios. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 8215–8226. [CrossRef] [PubMed]
3. Kim, M.; Choi, M.S.; Jang, G.R.; Bae, J.H.; Park, H.S. EEG-controlled tele-grasping for undefined objects. *Front. Neurorobot.* **2023**, *17*, 1293878. [CrossRef] [PubMed]
4. Wan, G.; Chen, J.; Zhang, J.; Liu, B.; Zhang, H.; Tao, X. Research on robot monocular vision-based 6DOF object positioning and grasping approach combined with image generation technology. *IEEE Access* **2023**, *11*, 136910–136923. [CrossRef]
5. Goberville, N.; El-Yabroudi, M.; Omwanas, M.; Rojas, J.; Meyer, R.; Asher, Z.; Abdel-Qader, I. Analysis of LiDAR and camera data in real-world weather conditions for autonomous vehicle operations. *SAE Int. J. Adv. Curr. Pract. Mobil.* **2020**, *2*, 2428–2434. [CrossRef]

6. Wan, G.; Wang, G.; Xing, K.; Fan, Y.; Yi, T. Robot visual measurement and grasping strategy for roughcastings. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 1729881421999937. [CrossRef]
7. Yin, Z.; Li, Y. Overview of robotic grasp detection from 2D to 3D. *Cogn. Robot.* **2022**, *2*, 73–82. [CrossRef]
8. Yang, J.Y.; Chen, U.K.; Chang, K.C.; Chen, Y.J. A novel robotic grasp detection technique by integrating YOLO and grasp detection deep neural networks. In Proceedings of the 2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, Taiwan, China, 19–21 August 2020; pp. 1–4. [CrossRef]
9. Sun, R.; Wu, C.; Zhao, X.; Zhao, B.; Jiang, Y. Object recognition and grasping for collaborative robots based on vision. *Sensors* **2024**, *24*, 195. [CrossRef] [PubMed]
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
11. Xiao, Y.; Tian, Z.; Yu, J.; Zhang, Y.; Liu, S.; Du, S.; Lan, X. A review of object detection based on deep learning. *Multimed. Tools Appl.* **2020**, *79*, 23729–23791. [CrossRef]
12. El-Yabroudi, M.Z.; Abdel-Qader, I.; Bazuin, B.J.; Abudayyeh, O.; Chabaan, R.C. Guided depth completion with instance segmentation fusion in autonomous driving applications. *Sensors* **2022**, *22*, 9578. [CrossRef] [PubMed]
13. Jin, M.; Li, J.; Zhang, L. DOPE++: 6D pose estimation algorithm for weakly textured objects based on deep neural networks. *PLoS ONE* **2022**, *17*, e0269175. [CrossRef] [PubMed]
14. Chen, Y.L.; Cai, Y.R.; Cheng, M.Y. Vision-based robotic object grasping—A deep reinforcement learning approach. *Machines* **2023**, *11*, 275. [CrossRef]
15. Qi, W.; Fan, H.; Zheng, C.; Su, H.; Alfayad, S. Human-like dexterous grasping through reinforcement learning and multimodal perception. *Biomimetics* **2025**, *10*, 186. [CrossRef] [PubMed]
16. Muratore, F.; Ramos, F.; Turk, G.; Yu, W.; Gienger, M.; Peters, J. Robot learning from randomized simulations: A review. *Front. Robot. AI* **2022**, *9*, 799893. [CrossRef] [PubMed]
17. Du, K.; Song, J.; Wang, X.; Li, X.; Lin, J. A multi-object grasping detection based on the improvement of YOLOv3 algorithm. In Proceedings of the 2020 Chinese Control and Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 1027–1033. [CrossRef]
18. Li, L.; Li, N.; Nan, R.; He, Y.; Li, C.; Zhang, W.; Fan, P. Robotic grasping technology integrating large kernel convolution and residual connections. *Machines* **2024**, *12*, 786. [CrossRef]
19. Song, Y.; Wen, J.; Liu, D.; Yu, C. Deep robotic grasping prediction with hierarchical RGB-D fusion. *Int. J. Control Autom. Syst.* **2022**, *20*, 243–254. [CrossRef]
20. Yu, T.; Bidulka, L.; McKeown, M.J.; Wang, Z.J. PA-tran: Learning to estimate 3D hand pose with partial annotation. *Sensors* **2023**, *23*, 1555. [CrossRef] [PubMed]
21. Rohmer, E.; Singh, S.P.N.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 1321–1326. [CrossRef]
22. Hou, Q.; Zhou, D.; Feng, J. Coordinate attention for efficient mobile network design. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13713–13722. [CrossRef]
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
24. Lu, S.; Lu, H.; Dong, J.; Wu, S. Object detection for UAV aerial scenarios based on vectorized IoU. *Sensors* **2023**, *23*, 3061. [CrossRef] [PubMed]

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