Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

ProtoRefine: Prototype-conditioned refinement of strawberry instance masks with native-area decoding

Minhui Kang

zebukang@hotmail.com

Jiangxi Electric Vocational and Technical College

Yiyao Zhang

Jiangxi Electric Vocational and Technical College

Jiayun Zheng

Jiangxi Electric Vocational and Technical College

Peizhe Long

Shandong Agricultural University

Jiafeng Qi

Jiangxi Electric Vocational and Technical College

Research Article

Keywords: strawberry; instance segmentation; local refinement; prototype conditioning; native-area decoding; agricultural vision

Posted Date: November 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11441508/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

ProtoRefine: Prototype-conditioned refinement of strawberry instance masks with native-area decoding

Abstract

In strawberry canopy images, leaves and stems can obscure fruit contours, leaving correctly localized instances with masks that include vegetation or miss visible fruit regions. We propose ProtoRefine to correct these post-detection errors by reusing the instance representations of a prototype-based segmenter, without an additional image encoder. A shared local predictor combines spatial prototypes, instance coefficients, coarse mask logits and coordinates to predict foreground scores. Native-area decoding then preserves the native mask's foreground count and reallocates pixel locations according to these scores, while leaving boxes, classes and confidence scores unchanged. On the official StrawDI test set at 640-pixel input, ProtoRefine achieves 84.03 mask average precision (AP), improving native YOLO26s-seg by 1.32 percentage points in AP and 2.90 points in AP90, with only 25,144 additional trainable parameters. Controlled comparisons across three local-head initializations show that the additional AP benefit of explicit prototype factors over coarse-field inputs depends on decoding and is larger under free thresholding. Matched-instance analyses show higher mean intersection over union (IoU) and boundary F1 than native masks at unchanged foreground count, while an inaccurate native count limits correction. These results show that existing instance representations can support compact local mask correction, and that the incremental benefit of explicit prototype inputs depends on mask reconstruction.

Keywords: strawberry; instance segmentation; local refinement; prototype conditioning; native-area decoding; agricultural vision

1 Introduction

Instance masks delineate the visible region of individual fruits for applications such as image-based size estimation and ripeness analysis (Giménez-Gallego et al. 2024; Zhao et al. 2025). In strawberry canopies, leaves, stems and neighboring fruits can obscure contours, leaving irregular outlines or small exposed regions (Pérez-Borrero et al. 2020). Even when a detector correctly localizes a fruit, its mask may include

surrounding vegetation or miss visible fruit pixels. Correcting these errors requires refining foreground–background assignments around an already localized instance.

Local refinement can revisit poorly delineated regions using representations already computed by a segmenter. Prototype-based systems combine spatial prototypes and instance coefficients into a coarse foreground field (Bolya et al. 2019; Chen et al. 2020). These factors remain available, but already contribute to the composed field. Their additional value as explicit inputs therefore requires a controlled comparison. Binary reconstruction raises a related question: does this value depend on how predicted scores are converted into a mask?

We introduce ProtoRefine, a compact local refinement branch on the native YOLO26s-seg mask path. A shared convolutional predictor combines spatial prototypes, instance coefficients, coarse logits and coordinates on aligned local supports. Training updates only this predictor, without an additional image encoder. At inference, predefined routing rules select small-instance masks for replacement; boxes, classes, confidence scores and candidate order remain unchanged. Learning thus focuses on mask correction within existing detections.

Binary reconstruction determines foreground quantity and location. Free thresholding allows both to change. Our native-area decoder inherits the native mask's foreground count and selects the highest-scoring admissible pixels using the locally predicted field. This separates spatial reallocation at fixed count from changes in foreground quantity. The inherited count also limits attainable quality: spatial rearrangement cannot remove the quantity error when the native mask overestimates or underestimates visible foreground.

On the official StrawDI test set at 640-pixel input, ProtoRefine achieves 84.03 mask AP, improving native YOLO26s-seg by 1.32 percentage points in AP and 2.90 points in AP90 with 25,144 additional trainable parameters. Development comparisons use complete systems as cross-architecture references, while input and capacity controls on a fixed detector examine the refinement branch. Across three local-head initializations, explicit prototype factors provide larger AP gains over coarse-field inputs under free thresholding than under native-area decoding. Matched-instance analyses further show higher mean overlap and boundary quality than native masks at unchanged foreground count, while identifying cases where native count errors limit correction.

The main contributions are as follows:

1. **Compact refinement from native instance representations.** ProtoRefine combines spatial prototypes, instance coefficients and coarse-mask evidence in a shared local predictor to refine detected strawberry masks without an additional image encoder or changes to detection outputs.
2. **Reconstruction with the native foreground count.** Native-area decoding preserves the existing mask's foreground count and reallocates pixel locations using learned scores, providing an explicit reconstruction rule for correction at fixed count.
3. **Representation benefits and decoding dependence.** Input and capacity controls, together with three local-head initializations, test the additional value of explicit prototype factors. Matched-instance geometry characterizes fixed-count spatial improvements and limitations from inherited count errors.

2 Related work

2.1 Strawberry instance segmentation

Strawberry instance segmentation delineates individual visible fruit regions among leaves, stems and neighboring fruits. StrawDI provides a public benchmark for this task (Pérez-Borrero et al. 2020). Existing approaches use centroid-based instance grouping (Perez-Borrero et al. 2021), lightweight multiscale mask prediction (Cao et al. 2023) and detection-initialized contour evolution (Guo et al. 2024). TensorRT has also been used to accelerate Mask R-CNN inference for strawberry perception (Crespo et al. 2025). These studies advance instance organization, mask construction and deployment efficiency.

Fruit masks also support image-based size estimation (Giménez-Gallego et al. 2024) and in-field ripeness analysis (Zhao et al. 2025), while foundation-model supervision can reduce annotation effort (Wang et al. 2026). ProtoRefine addresses visible-mask errors within existing detections, using native segmentation representations to correct included background and omitted visible fruit. We evaluate these corrections through mask overlap and boundary agreement.

2.2 Shared representations and instance conditioning

Instance segmentation associates shared image representations with individual objects through different forms of conditioning. YOLACT uses spatial prototypes and instance coefficients (Bolya et al. 2019), BlendMask combines bottom-up bases with top-down attention (Chen et al. 2020), and CondInst employs instance-conditioned filters and relative coordinates (Tian et al. 2020). These methods provide different implementations of the association between shared features and a target instance.

ProtoRefine uses the detector's existing instance representations at the local refinement stage, with convolutional weights shared across instances. The coarse mask already reflects prototype–coefficient composition, so the utility of exposing these factors explicitly must be assessed against coarse-field inputs. Input and capacity controls examine their additional value and its relation to binary reconstruction.

2.3 Mask refinement and reconstruction

Mask refinement improves spatial detail through further prediction in difficult regions or the use of fine-grained features. PointRend predicts selected uncertain points (Kirillov et al. 2020), RefineMask progressively integrates fine-grained features (Zhang et al. 2021), and Mask Transfiner organizes error-prone regions with a quadtree (Ke et al. 2022). Other approaches use global–local processing, boundary-to-interior correspondences or discrete diffusion (Cheng et al. 2020; Yuan et al. 2020; Wang et al. 2023). HQ-SAM illustrates a compact learned extension that reuses a frozen segmenter's representations to improve mask details (Ke et al. 2023). ProtoRefine performs local refinement on a prototype-based instance segmenter, directly using native instance representations to predict dense foreground scores.

Foreground-score prediction and binary reconstruction are linked stages. Explicit area constraints have been studied in segmentation (Niethammer and Zach 2013);

here, each native instance mask supplies a foreground-count budget, and local scores rank pixels for reconstruction. We therefore examine both the role of representation inputs and spatial changes at fixed count. Dynamic mask-resolution selection (Li et al. 2023) and mask-quality scoring (Huang et al. 2019) concern computational allocation and confidence estimation, while ProtoRefine preserves native detection outputs. Contour-sensitive evaluation, including Boundary IoU (Cheng et al. 2021), motivates complementing whole-system AP with matched-instance overlap and boundary F1.

3 Method

3.1 Overview and problem formulation

ProtoRefine refines foreground assignments within detected strawberry instances to reduce included background and omitted visible fruit (Figure 1). Given native YOLO26s-seg predictions, it selects small instances using predicted box area and native confidence, then predicts local foreground logits within expanded box supports. Native-area decoding preserves each native mask's foreground count and uses the new logits to reallocate foreground locations.

Only selected masks are replaced; other candidates retain their native masks. Candidate identities, ordering, boxes, classes and confidence scores remain unchanged. Training fixes the native network and optimizes only the shared local predictor. ProtoRefine denotes this complete pipeline, including native-area decoding. Routing thresholds and candidate limits are specified in Supplementary Section 9.

3.2 Prototype-conditioned local representation

The native mask path combines spatial prototypes into a scalar foreground estimate for each instance. Let $P(x) \in \mathbb{R}^{32}$ denote the shared prototype features at location x, and let $c_i \in \mathbb{R}^{32}$ denote the coefficients for instance i. The native coarse logit field is

$$z_i(x) = c_i^{\mathsf{T}} P(x). \tag{1}$$

ProtoRefine retains both this estimate and its constituent representations. Explicit access to P and c_i allows local convolutions to combine neighboring prototype responses with the target instance's coefficients, while z_i provides the initial foreground estimate. Coarse-input controls in Section 5 test the additional utility of these factors under each decoding rule.

For each selected instance, the predicted box is expanded by a factor of 1.25 around its center and clipped to the image. Prototypes and coarse logits are bilinearly sampled on the same 64×64 pixel-center grid. Coefficients are broadcast over the grid, and two normalized coordinate maps specify local position. At $u = (u_x, u_y)$, the input is

$$X_i(u) = [P_i(u), c_i, u_x, u_y, z_i(u)] \in \mathbb{R}^{67}. \tag{2}$$

Here $P_i(u)$ and $z_i(u)$ are the sampled prototype features and coarse logits. Coordinates locate each sample within the support. Figure 2 shows the alignment and concatenation operations; exact sampling conventions are given in Supplementary Section 9.

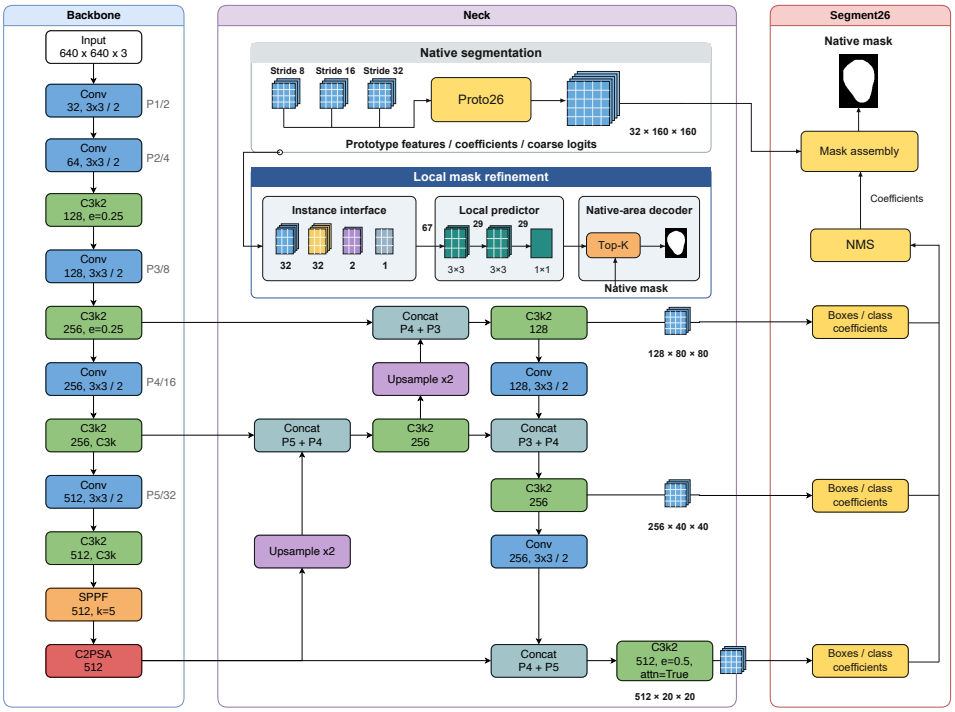


Fig. 1 Overall architecture. Native YOLO26s-seg supplies prototypes, coefficients, coarse logits and masks to the local branch. Routed masks are replaced; other candidates retain their native predictions. Channel groups identify the interface, predictor and decoder. Backbone and neck topology follows the YOLO26 reference (Jocher et al. 2026), with s-scale dimensions. Feature planes are schematic; mask thumbnails are training-instance predictions.

3.3 Shared dense local prediction and learning

All instances use the same convolutional weights; instance conditioning enters through the aligned inputs. Two 3×3 convolutions with ReLU activation combine local spatial neighborhoods, followed by a 1×1 projection. Channel dimensions are $67 \to 29 \to 29 \to 1$, and all layers preserve the 64×64 grid. The resulting foreground logit field uses a fixed scale of three:

$$q_i = 3 f_\theta(X_i). \tag{3}$$

The predictor contains 25,144 trainable parameters. It uses the fixed network's representation and local coordinates, with no additional image encoder.

Training candidates are matched to annotated instances using bounding-box IoU. A best match of at least 0.5 supplies the corresponding mask target; candidates below 0.1 receive an all-background target, and intermediate candidates are ignored. Binary cross-entropy is averaged over the local grid. Positive candidates additionally receive an equally weighted soft Dice loss with a smoothing constant of one, and the combined loss is averaged over instances.

The same trained predictor is evaluated with two reconstruction rules: free decoding thresholds the rendered logits at zero, while native-area decoding selects pixels by rank at the native foreground count. Comparing these outputs holds the learned field fixed and isolates the reconstruction rule; both use the segmentation-trained predictor without further optimization.

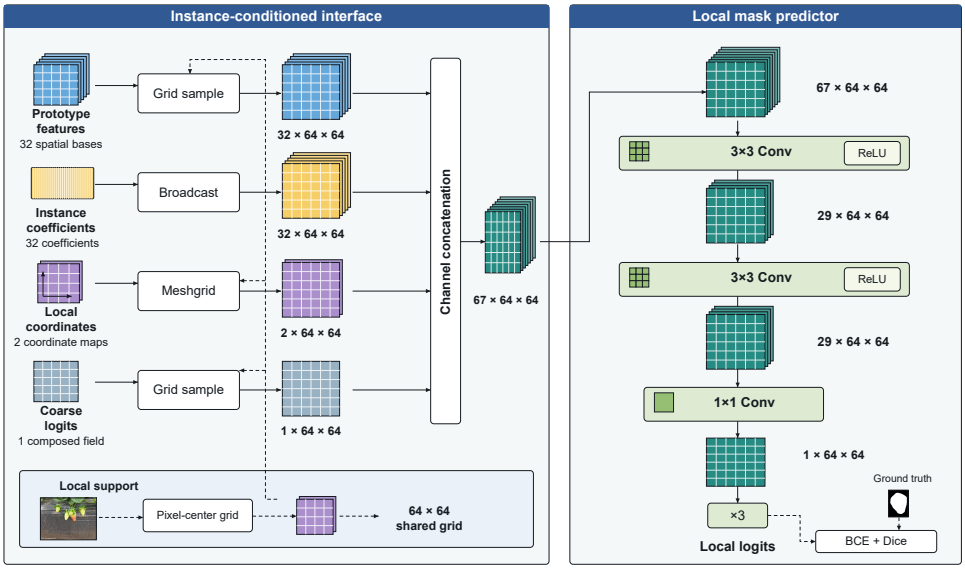


Fig. 2 Instance-conditioned interface and local predictor. Sampling, coefficient broadcasting and coordinates form a 67-channel input on a shared 64×64 grid. Two 3×3 convolutions and a 1×1 projection produce local logits. Solid arrows show feature flow; dashed arrows show grid or supervision flow. Plane depths are schematic. RGB locates the support; ground truth supplies training supervision.

3.4 Native-area decoding

Converting local logits into a binary mask determines both foreground quantity and location. Free thresholding allows both to change. Native-area decoding instead inherits the native mask's foreground count and uses the learned logits to determine which pixels occupy that budget.

Let N_i be the native binary mask in original-image coordinates and $K_i = |N_i|$ its foreground count. Bilinear interpolation renders q_i onto the original-image support as $\tilde{q}_i$, with scores outside the support set to negative infinity. The admissible domain Ω_i is the union of that support and the native foreground. Reconstruction selects

$$M_i = \mathrm{TopK}(\tilde{q}_i, \Omega_i, K_i), \qquad |M_i| = K_i = |N_i|. \tag{4}$$

Including the native foreground ensures that the admissible domain contains at least K_i pixels. Figure 3 illustrates reconstruction at unchanged count. Supplementary

Section 6.3 specifies tie handling, empty masks and cases where the local support contains fewer than K_i pixels.

For finite admissible scores, top-K maximizes their selected sum subject to the count constraint. With distinct scores across the cutoff, selection is equivalent to an instance-specific threshold set by K_i. The inherited quantity is the native prediction's pixel count, rather than the annotated or physical fruit area.

At fixed foreground count, replacing an incorrectly included background pixel with a missed fruit pixel reduces both false positives and false negatives by one. More generally, for a fixed annotation G_i, any mask A satisfies $FP(A) - FN(A) = |A| - |G_i|$. Since $|M_i| = |N_i|$,

$$FP(M_i) - FP(N_i) = FN(M_i) - FN(N_i). \tag{5}$$

Thus, changes in false foreground and missed foreground are equal, allowing spatial reallocation to be examined at a fixed predicted count. Any discrepancy between native and annotated foreground counts remains and limits attainable overlap. Boundary quality is evaluated separately because the identity does not specify where the remaining errors occur.

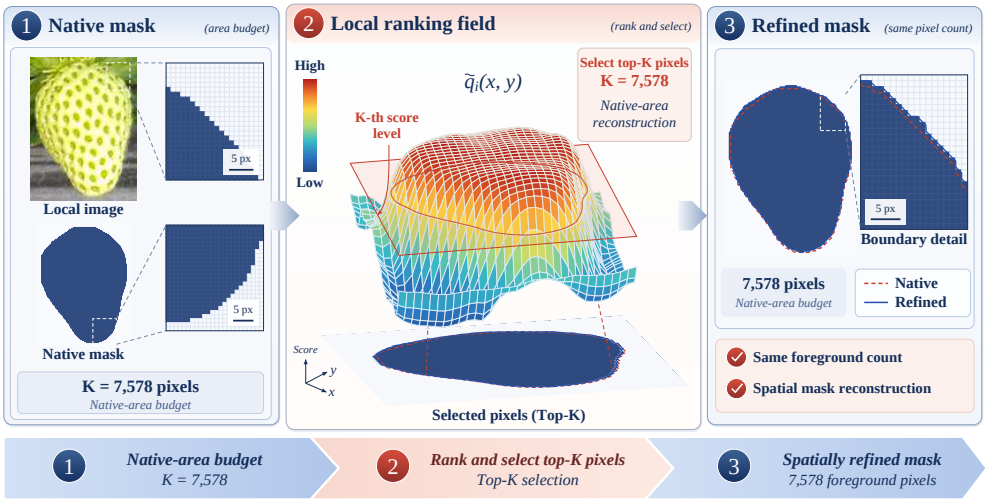


Fig. 3 Native-area decoding. The native mask supplies the foreground count (7578 pixels in this training example). The saved rendered field $\tilde{q}_i$ is downsampled and height-normalized for display. The red contour is computed at the K-th score; the footprint and refined mask show the discrete top-K selection. Insets use five-original-pixel scale bars.

4 Experimental setup

4.1 Dataset and data partition

We evaluate single-class strawberry instance segmentation on StrawDI Db1 (Pérez-Borrero et al. 2020), which contains 3100 annotated images from production scenes with varying fruit scales and canopy occlusion. Its official training, validation and test partitions contain 2800, 100 and 200 images, respectively.

From the official training partition, filename-index blocks and a neighboring-index exclusion buffer define 2427 training images and 265 development images, with 108 images excluded as a buffer. The development set contains 1471 instances and is separate from the official validation set. Final evaluation uses all 200 official test images and their 1132 annotated instances. Filename grouping does not establish scene- or plantation-independent splits; the construction and data roles are detailed in Supplementary Section 7.1.

The official test evaluates the prespecified models and comparisons. Subsequent component analyses and additional system comparisons use the internal development set, without altering those test models. Development and test results are reported separately.

4.2 Implementation and training

The native model is single-class YOLO26s-seg, implemented in Ultralytics 8.4.173 (Jocher et al. 2026). After detector training, the native network is fixed and its cached representations are used to train the local head. Head training uses 12 epochs, initialization seed 42, instance batch size 32 and AMP. AdamW uses weight decay 10^{-4} and a cosine learning-rate schedule from 10^{-3} to 10^{-4}. We use the final head without EMA or development-based checkpoint selection.

At 640-pixel input, refinement selects candidates with native score ≥ 0.05 and positive predicted box area ≤ 4096 pixels. At most 16 candidates per image are selected in descending native-score order, with stable ties. Support construction and local sampling follow Section 3.2. Supplementary Table S11 and the model-specific recipes give the remaining implementation settings.

4.3 System comparisons and component studies

Complete-system comparisons assess segmentation quality relative to native YOLO26s-seg and alternative architectures. Table 1 separates official-test and development results; all systems use the same 2427-image training partition. YOLO11s-seg (Ultralytics n.d.), RTMDet-Ins-tiny and RTMDet-Ins-s (Lyu et al. 2022), Mask R-CNN R50-FPN (He et al. 2017), and YOLACT–ResNet50-FPN (Bolya et al. 2019) provide cross-architecture references. Native YOLO26s-seg directly measures the gain from refinement on the same detector, while 896-pixel inference provides a higher-resolution quality reference. Each family retains its own resizing, training and inference procedures. Parameter counts cover complete single-class inference models, counting shared weights once. Full metrics and literature-reported results under their original settings appear in Supplementary Tables S1, S12 and S13.

The three added systems—Mask R-CNN, RTMDet-Ins-s and YOLACT—are evaluated on development data. They use full COCO initialization, seed 42, a declared 640-pixel setting and 60 training epochs, retaining final weights (final EMA for RTMDet). Family-specific recipes are given in Supplementary Section 8. The main analysis uses one fixed YOLO26 detector throughout; a second fixed detector provides additional comparisons in the Supplementary Material.

To test whether the value of explicit prototype factors depends on reconstruction, we cross coarse/full inputs with free/native-area decoding. Coarse input contains logits and coordinates; full input adds prototypes and coefficients. A wider coarse head has 24,940 parameters, close to the full head's 25,144, to control the parameter budget. Both heads use the same training instances, epoch sample orders, optimizer and duration, with initialization seeds 7, 42 and 123. Only the local heads are retrained; the detector stays fixed, and seed 42 is the main setting. Widths and initialization draws differ. Each trained head is evaluated with both decoding rules.

To compare local prediction strategies, we include a 25,442-parameter PointRend-inspired adaptation (Kirillov et al. 2020). It provides a point-prediction reference to dense local refinement, with its sampling, supervision and rendering conventions specified in Supplementary Section 6.1.

4.4 Evaluation metrics and statistical analysis

Complete-system quality is measured by COCO mask AP over IoU thresholds from 0.50 to 0.95, together with AP90 for strict overlap and size-specific AP for small, medium and large instances. AP50 and AP75 are provided in the Supplementary Material. Scores use the 0–100 scale; differences are percentage points (pp).

Geometric analysis isolates mask changes within detected instances. Native-box correspondences are fixed, and native masks, free decoding and native-area decoding are compared on the same 864 matched development instances selected for refinement. We report instance-averaged mask IoU and boundary F1 with a two-original-pixel tolerance. Omission and spill are false-negative and false-positive pixel counts divided by each instance's annotated foreground area. These conditional diagnostics exclude missed or unmatched objects and complement whole-system AP.

Official-test comparisons use 1000 paired image-bootstrap draws, with common sampled images across methods and global AP recomputed for each draw. The 95% percentile intervals describe image-sampling uncertainty conditional on fixed models, rather than training variability. The three local-head initializations are summarized descriptively. Further statistical details, including development comparisons, are given in Supplementary Section 7.3.

5 Results

5.1 Improving strawberry delineation after detection

Refining the masks of existing detections improves whole-system segmentation quality. On the official StrawDI test set, ProtoRefine reaches 84.03 mask AP at 640-pixel input, a gain of 1.32 percentage points over native YOLO26s-seg, with a paired-image 95%

interval of [1.02, 1.51] (Table 1(a)). Bounding boxes, confidence scores and candidate ordering remain unchanged, attributing the difference to the revised masks.

Method	Type	Backbone	Params (M)	Input (px)	AP	AP90	AP_S
(a) Official test: 200 images, 1132 instances							
RTMDet-Ins-tiny	Dynamic	CSPNeXt-tiny	5.615	640	80.10	64.17	38.65
YOLO11s-seg	Prototype	YOLO11-s	10.083	640	82.28	65.48	45.88
Native YOLO26s-seg	Prototype	YOLO26-s	11.434	640	82.72	66.36	47.14
ProtoRefine (Ours)	Prototype+local	YOLO26-s	11.460	640	**84.03**	**69.26**	**49.18**
Higher-resolution quality references							
RTMDet-Ins-tiny	Dynamic	CSPNeXt-tiny	5.615	896	83.60	68.51	47.44
Native YOLO26s-seg	Prototype	YOLO26-s	11.434	896	84.81	69.44	51.93
(b) Shared development split: 265 images, 1471 instances							
YOLACT	Prototype	ResNet-50	34.727	640	76.37	52.72	38.28
Mask R-CNN	Region	ResNet-50	43.971	640	78.58	61.37	37.73
RTMDet-Ins-tiny	Dynamic	CSPNeXt-tiny	5.615	640	80.56	61.36	42.26
RTMDet-Ins-s	Dynamic	CSPNeXt-s	10.153	640	81.56	63.27	44.06
YOLO11s-seg	Prototype	YOLO11-s	10.083	640	83.03	64.43	50.32
Native YOLO26s-seg	Prototype	YOLO26-s	11.434	640	83.13	64.34	50.09
ProtoRefine (Ours)	Prototype+local	YOLO26-s	11.460	640	**84.52**	**68.40**	**52.40**

Table 1 Complete-system comparisons on StrawDI. Bold marks the best values within each 640-pixel block.

The improvement extends to stricter overlap: AP90 increases by 2.90 points, alongside higher small-object AP (Figure 4). Mask correction therefore improves the evaluated predictions without changing detection outputs or increasing the input resolution.

5.2 Mask quality across segmentation systems

ProtoRefine has the highest overall mask AP among the evaluated systems on the shared development split (Table 1(b)). Differences are clearer in small-object AP and AP90 than in large-object AP (Supplementary Table S12). On the official test, it also exceeds the evaluated YOLO11s-seg and RTMDet-Ins-tiny pipelines at the same input setting (Table 1(a)).

The examples in Figure 5 illustrate why local contour placement matters. In (a), smoothing over the concavity created by an occluding leaf includes vegetation within the fruit mask. In (b), a crossing stem creates a separation that the visible mask should preserve. In (c), a change in predicted extent alters the silhouette of a small fruit. Several competing predictions fill these structures or distort the outline, whereas ProtoRefine follows the annotated visible contours more closely in these cases.

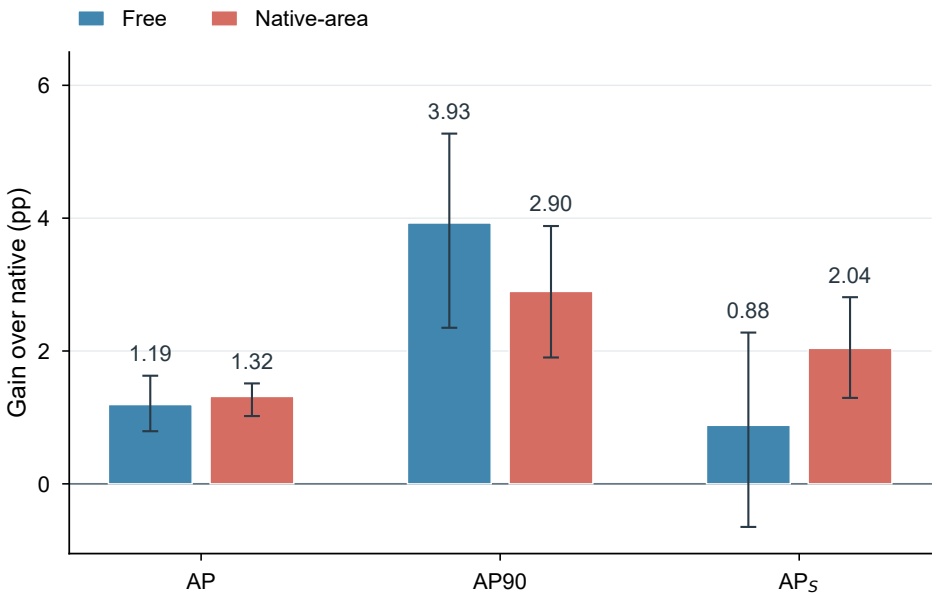


Fig. 4 Within-base mask refinement on the official test. Gains over native YOLO26s-seg at 640 pixels. Whiskers are 95% paired-image intervals conditional on the fixed reference checkpoint.

The native YOLO26s-seg masks already provide a strong reference. Refinement raises development AP90 by 4.06 points while retaining the same detections, identifying a gain on this fixed detector. Cross-system differences additionally reflect backbones, pretraining and training recipes; the selected examples illustrate contour behavior rather than aggregate performance for specific occlusion types.

5.3 Explicit instance evidence and mask reconstruction

Explicit prototype factors contribute differently depending on how local logits are reconstructed into masks. We compare full and near-capacity-matched coarse heads under free thresholding and native-area decoding. Table 2 gives the main seed-42 configuration; Supplementary Table S4 repeats the complete comparison at seeds 7, 42 and 123 with the detector fixed.

Across all three head initializations, the full-input AP gain is larger under free decoding. Its mean is 0.53 points with free decoding and 0.08 points with native-area decoding. Under the count constraint, the full-input advantage changes sign across runs. Thus, the additional value of exposing prototype and coefficient inputs is conditional on reconstruction, rather than a consistent gain of similar size under both rules.

Free decoding depends on logits relative to zero, whereas native-area decoding uses their ordering at a prescribed count. This distinction offers a possible explanation for the different input gains. However, AP90 does not show a consistent interaction direction, and head widths and optimization trajectories also differ; the evidence does not

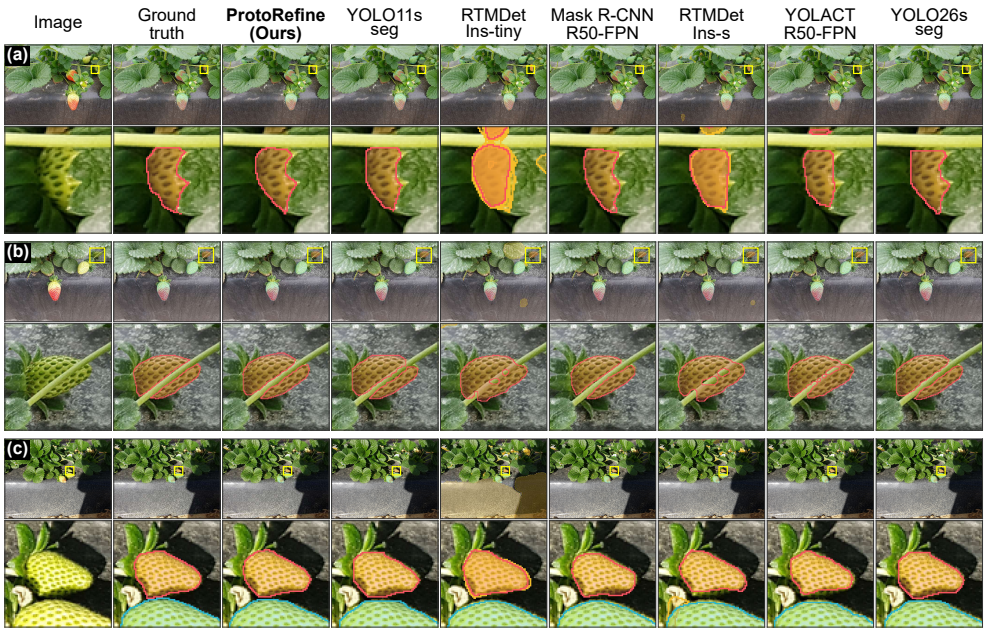


Fig. 5 Selected visible-contour improvements. (a) A leaf-occluded fruit; (b) a fruit crossed by a stem; (c) a small fruit. Scene/detail pairs use identical crops across all seven systems, with ProtoRefine beside ground truth. Coral marks the target, cyan neighboring display-matched instances and amber unmatched candidates; amber is not an error label. Yellow boxes locate crops. These favorable examples illustrate local differences; display and selection details appear in Supplementary Section 10.

(a) Component configurations

Configuration	Local predictor	Full interface	Area decoder	Added params	AP	AP90	AP small	ΔAP
Native	–	–	–	0	83.1275	64.3380	50.0880	+0.0000
Coarse local	✓	–	–	24,940	83.3828	67.3781	47.2848	+0.2554
Full local	✓	✓	–	25,144	83.9342	67.4457	49.5083	+0.8067
Coarse + area	✓	–	✓	24,940	84.3820	68.4222	51.8755	+1.2546
Full + area	✓	✓	✓	25,144	84.5238	68.4022	52.4034	+1.3963

(b) Conditional component gains

Comparison	ΔAP	ΔAP90	ΔAP small
Local prediction with coarse inputs	+0.2554	+3.0401	-2.8033
Explicit factors under free decoding	+0.5513	+0.0676	+2.2235
Area decoder with the full interface	+0.5896	+0.9566	+2.8951
Area decoder with coarse inputs	+0.9992	+1.0441	+4.5908
Explicit factors under area decoding	+0.1418	-0.0199	+0.5278

Table 2 Component ablation on the development set for the main head seed 42. (a) Configurations and their AP change from native prediction. The full interface adds explicit prototype/coefficient factors. (b) Conditional gains in pp. The three-initialization comparison is in Supplementary Table S4.

isolate calibration as the mechanism. The PointRend-inspired adaptation likewise has nearly the same development AP as free dense prediction (Supplementary Table S6), so these results do not establish an advantage of dense prediction alone.

The test comparison further distinguishes the complete refinement gain from the count constraint's incremental effect. Native-area versus free decoding gives a small positive AP difference for the main detector and a small negative difference for the second, with both paired intervals spanning zero (Supplementary Table S3). Local refinement improves on native masks, but the test results do not establish a consistent additional AP benefit from fixing foreground count.

5.4 Foreground placement and visible-boundary quality

Figure 6 illustrates spatial correction at an occluding leaf, near the calyx and along a concave fruit–vegetation boundary. In these examples, foreground outside the annotation is removed and assigned to missed fruit regions elsewhere. Because the total count is unchanged, the differences show where the native foreground budget is reallocated.

This improvement is also present in the averages over the same 864 development instances matched using fixed native boxes and selected for refinement. Mean boundary F1 rises from 87.95% to 89.77%, together with higher mean mask IoU (Table 3). These measurements assess the masks of matched instances, separately from whole-system AP.

Mask configuration	Geometric quality		Foreground errors	
	Mask IoU ↑	Boundary F1 ↑	Omission ↓	Spill ↓
Native mask	86.15	87.95	5.98	10.20
Free local prediction	85.72	88.41	9.67	**5.80**
Native-area decoding	**87.28**	**89.77**	**5.29**	9.51

Table 3 Geometric quality and foreground errors. Means (%) on the same 864 routed native-box matches in the development set. Boundary F1 uses a two-original-pixel tolerance; omission and spill are FN and FP normalized by annotated foreground area.

Relative to native masks, omission and spill both decrease. Their equal reductions follow from the fixed-count identity in Equation 5; the measured reduction establishes better foreground placement on average. The comparison with free decoding reveals a different trade-off. Free prediction removes more spill but also misses more visible fruit, raising mean boundary F1 while lowering mean IoU relative to native masks. Native-area decoding reduces this omission at the cost of retaining more spill than free decoding.

The instance distributions in Figure 7 distinguish average gains from how often individual masks improve. Relative to native masks, IoU improves for most instances. Relative to free decoding, slightly more masks lose IoU than gain it, and the median

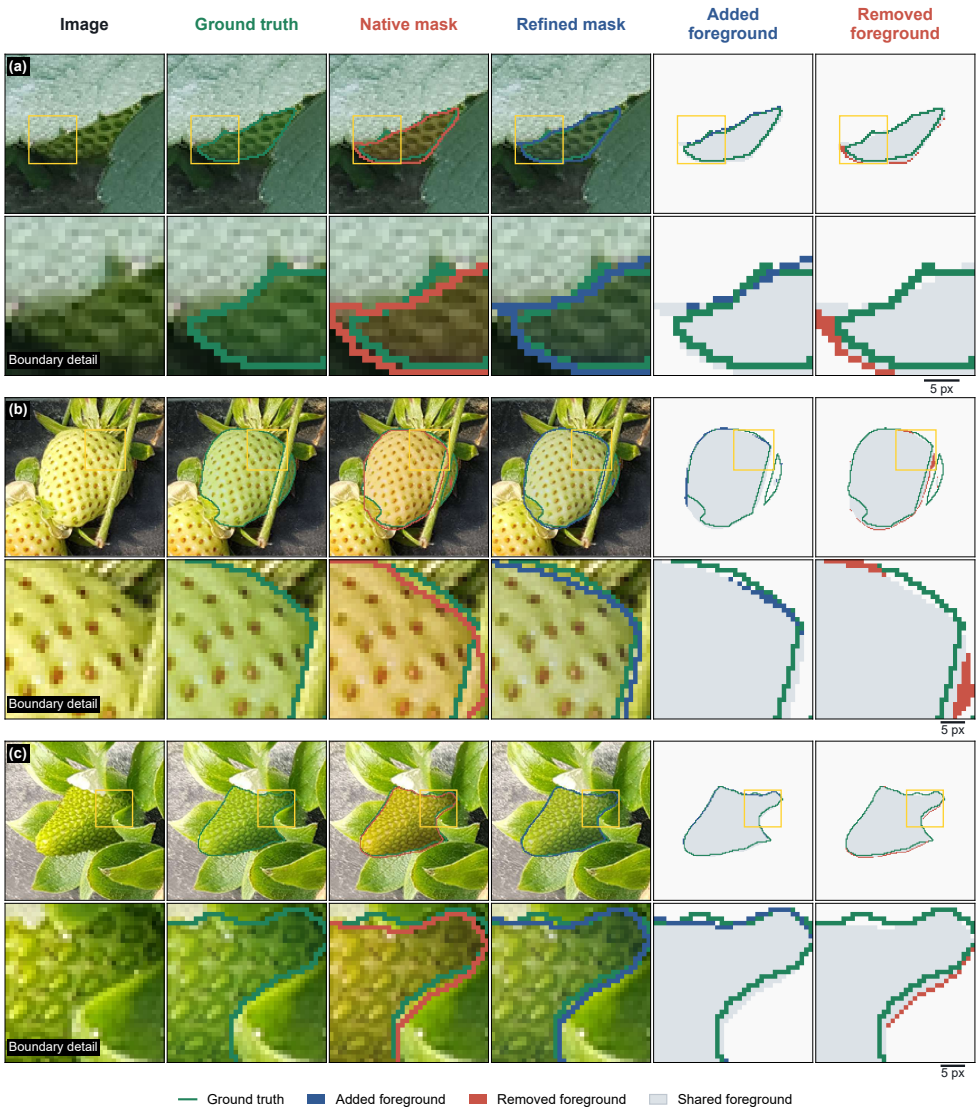
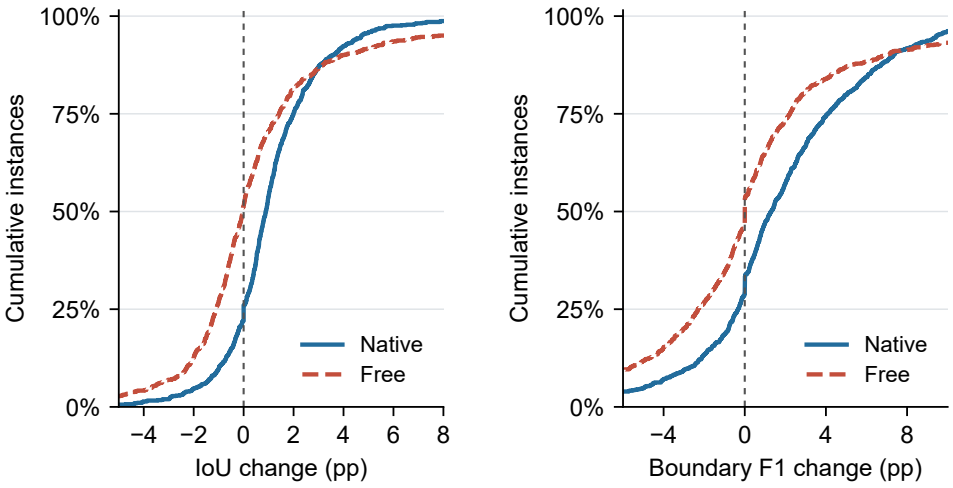


Fig. 6 Boundary detail and foreground redistribution. Selected development cases show (a) an occluded fruit, (b) the calyx and (c) a concave leaf boundary. All six columns share the enlarged window marked in yellow. Blue and coral show added and removed foreground, gray shared foreground and green the annotated contour. Native-area decoding preserves the foreground count. Scale bars denote five original-image pixels.

boundary F1 change is zero. The positive mean changes against free decoding therefore coexist with mixed individual outcomes.

The limitation is particularly visible when the native count overestimates annotated foreground. In both strata exceeding the annotated area by more than 10%, native-area decoding has lower mean boundary F1 than free decoding (Supplementary



(a) Mask IoU (b) Boundary F1

Fig. 7 How refinement gains vary across instances. Changes in (a) mask IoU and (b) boundary F1 for ProtoRefine minus the indicated comparator on the same 864 routed development matches. The vertical line marks no change. Central horizontal ranges are shown, with all instances retained in each cumulative denominator; the complete ranges appear in Supplementary Figure S9. Boundary F1 uses a two-original-pixel tolerance.

Table S8). This pattern is consistent with an excessive inherited budget limiting the removal of surplus foreground. Additional failure and little-change examples in Supplementary Section 10 illustrate inherited quantity errors and unnecessary changes to already accurate outlines.

5.5 Refinement and higher-resolution inference

ProtoRefine adds 25,144 trainable parameters, approximately 0.22% of the native model, without an additional image encoder. Higher-resolution native inference nevertheless remains a strong quality reference: YOLO26s-seg at 896 pixels reaches 84.81 mask AP, compared with 84.03 for ProtoRefine at 640 pixels (Table 1(a)). Local refinement improves masks at the original input setting, but does not replace all of the quality gains available from increasing native input resolution.

6 Discussion

For strawberries that have already been localized, mask errors can still arise from included background and omitted visible fruit. In the evaluated setting, ProtoRefine shows that representations already computed by the segmenter can support further local correction without an additional image encoder or changes to detection outputs. Explicit prototypes and instance coefficients expose the factors underlying the coarse foreground estimate, but their additional utility is linked to reconstruction. Across three local-head initializations, the full-input AP advantage over coarse inputs is larger

under free decoding. In this study, the value of representation reuse therefore needs to be assessed together with how predicted scores are converted into binary masks.

Native-area decoding uses the existing mask's foreground count as a reconstruction budget and lets local scores reallocate its spatial support. Matched-instance analysis shows that this reallocation improves mean overlap and boundary quality at unchanged count. Relative to free decoding, it reduces mean omission while retaining more spill. Both effects reflect the same design trade-off: the native count supplies a budget but also restricts correction of quantity errors. When native foreground is overestimated, the observed boundary losses relative to free decoding are consistent with this restriction. A natural next question is whether prediction information alone can indicate when the native count is reliable enough to retain; the present method does not establish such an inference rule.

The evidence combines within-detector refinement comparisons on the official test with component and geometric analyses on internal development data. Three head initializations support the input–decoding relationship for the fixed detector, while its transfer to other backbones and a unique calibration mechanism remain unverified. The target is the annotated visible fruit region, rather than the complete shape behind an occluder. Improved mask quality does not by itself establish better size estimates or harvesting decisions, which require task-specific validation.

7 Conclusion

We introduced ProtoRefine to refine detected strawberry masks using a segmenter's existing spatial prototypes, instance coefficients and coarse foreground information. A shared local predictor and native-area decoding improve mask quality while preserving detection outputs, increasing mask AP by 1.32 percentage points over native YOLO26s-seg on the official StrawDI test. Controlled experiments further show that the additional AP value of explicit prototype factors depends on decoding. Retaining the native foreground count supports spatial correction while inheriting its quantity errors. These findings indicate that local refinement should consider both which representations to reuse and how their predicted scores are reconstructed into masks.

References

Bolya D, Zhou C, Xiao F, et al (2019) YOLACT: Real-time Instance Segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, URL https://openaccess.thecvf.com/content_ICCV_2019/papers/Bolya_YOLACT_Real-Time_Instance_Segmentation_ICCV_2019_paper.pdf

Cao L, Chen Y, Jin Q (2023) Lightweight Strawberry Instance Segmentation on Low-Power Devices for Picking Robots. Electronics 12(14):3145. https://doi.org/10.3390/electronics12143145, URL https://www.mdpi.com/2079-9292/12/14/3145

Chen H, Sun K, Tian Z, et al (2020) BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content_CVPR_2020/papers/Chen_BlendMask_Top-Down_Meets_Bottom-Up_for_Instance_Segmentation_CVPR_2020_paper.pdf

Cheng B, Girshick R, Dollár P, et al (2021) Boundary IoU: Improving Object-Centric Image Segmentation Evaluation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content/CVPR2021/papers/Cheng_Boundary_IoU_Improving_Object-Centric_Image_Segmentation_Evaluation_CVPR_2021_paper.pdf

Cheng HK, Chung J, Tai YW, et al (2020) CascadePSP: Toward Class-Agnostic and Very High-Resolution Segmentation via Global and Local Refinement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content_CVPR_2020/papers/Cheng_CascadePSP_Toward_Class-Agnostic_and_Very_High-Resolution_Segmentation_via_Global_and_CVPR_2020_paper.pdf

Crespo A, Moncada C, Crespo F, et al (2025) An efficient strawberry segmentation model based on Mask R-CNN and TensorRT. Artificial Intelligence in Agriculture 15(2):327–337. https://doi.org/10.1016/j.aiia.2025.01.008, URL https://www.sciencedirect.com/science/article/pii/S258972172500008X

Giménez-Gallego J, Martinez-del Rincon J, González-Teruel JD, et al (2024) On-tree fruit image segmentation comparing Mask R-CNN and Vision Transformer models. Application in a novel algorithm for pixel-based fruit size estimation. Computers and Electronics in Agriculture 222:109077. https://doi.org/10.1016/j.compag.2024.109077, URL https://pureadmin.qub.ac.uk/ws/portalfiles/portal/590267485/1-s2.0-S016816992400468X-main.pdf, institutional copy of the published article

Guo Z, Hu X, Zhao B, et al (2024) StrawSnake: A Real-Time Strawberry Instance Segmentation Network Based on the Contour Learning Approach. Electronics 13(16):3103. https://doi.org/10.3390/electronics13163103, URL https://www.mdpi.com/2079-9292/13/16/3103

He K, Gkioxari G, Dollár P, et al (2017) Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, URL https://arxiv.org/abs/1703.06870

Huang Z, Huang L, Gong Y, et al (2019) Mask Scoring R-CNN. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Huang_Mask_Scoring_R-CNN_CVPR_2019_paper.pdf

Jocher G, Qiu J, Liu M, et al (2026) Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. arXiv:2606.03748v1, URL https://arxiv.org/abs/2606.03748

Ke L, Danelljan M, Li X, et al (2022) Mask Transfiner for High-Quality Instance Segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content/CVPR2022/papers/Ke_Mask_Transfiner_for_High-Quality_Instance_Segmentation_CVPR_2022_paper.pdf

Ke L, Ye M, Danelljan M, et al (2023) Segment Anything in High Quality. In: Advances in Neural Information Processing Systems, URL https://papers.nips.cc/paper/2023/file/5f828e38160f31935cfe9f67503ad17c-Paper-Conference.pdf

Kirillov A, Wu Y, He K, et al (2020) PointRend: Image Segmentation as Rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content_CVPR_2020/papers/Kirillov_PointRend_Image_Segmentation_As_Rendering_CVPR_2020_paper.pdf

Li R, He C, Li S, et al (2023) DynaMask: Dynamic Mask Selection for Instance Segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content/CVPR2023/papers/Li_DynaMask_Dynamic_Mask_Selection_for_Instance_Segmentation_CVPR_2023_paper.pdf

Lyu C, Zhang W, Huang H, et al (2022) RTMDet: An Empirical Study of Designing Real-Time Object Detectors. arXiv:2212.07784, URL https://arxiv.org/abs/2212.07784

Niethammer M, Zach C (2013) Segmentation with area constraints. Medical Image Analysis 17(1):101–112. https://doi.org/10.1016/j.media.2012.09.002, URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3656501/

Pérez-Borrero I, Marín-Santos D, Gegúndez-Arias ME, et al (2020) A fast and accurate deep learning method for strawberry instance segmentation. Computers and Electronics in Agriculture 178:105736. https://doi.org/10.1016/j.compag.2020.105736, URL https://zenodo.org/records/14658929/files/PerezBorrero_et_al_2020_PostPrint.pdf?download=1, author accepted manuscript

Perez-Borrero I, Marin-Santos D, Vasallo-Vazquez MJ, et al (2021) A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network. Neural Computing and Applications 33:15059–15071. https://doi.org/10.1007/s00521-021-06131-2, URL https://link.springer.com/article/10.1007/s00521-021-06131-2

Tian Z, Shen C, Chen H (2020) Conditional Convolutions for Instance Segmentation. In: European Conference on Computer Vision, URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123460273.pdf

Ultralytics (n.d.) Ultralytics YOLO11. Official model documentation, URL https://docs.ultralytics.com/models/yolo11/, accessed 13 October 2026; publication date not specified

Wang M, Ding H, Liew JH, et al (2023) SegRefiner: Towards Model-Agnostic Segmentation Refinement with Discrete Diffusion Process. In: Advances in Neural Information Processing Systems, URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fc0cc55dca3d791c4a0bb2d8ddeefe4f-Paper-Conference.pdf

Wang Y, Fei Z, Li R, et al (2026) Learn from Foundation Model: Fruit Detection Model without Manual Annotation. arXiv:2411.16196v2, URL https://arxiv.org/pdf/2411.16196v2, version 2, revised 22 March 2026; first posted 25 November 2024. Associated journal DOI: 10.1016/j.patcog.2025.112799

Yuan Y, Xie J, Chen X, et al (2020) SegFix: Model-Agnostic Boundary Refinement for Segmentation. In: European Conference on Computer Vision, URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123570477.pdf

Zhang G, Lu X, Tan J, et al (2021) RefineMask: Towards High-Quality Instance Segmentation with Fine-Grained Features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, URL https://openaccess.thecvf.com/content/CVPR2021/papers/Zhang_RefineMask_Towards_High-Quality_Instance_Segmentation_With_Fine-Grained_Features_CVPR_2021_paper.pdf

Zhao Z, Hicks Y, Sun X, et al (2025) FruitQuery: A lightweight query-based instance segmentation model for in-field fruit ripeness determination. Smart Agricultural Technology 12:101068. https://doi.org/10.1016/j.atech.2025.101068, URL https://www.sciencedirect.com/science/article/pii/S2772375525003016

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- supplementary.pdf